JULY 1977 VOLUME 2, Number 7 $1.50 in USA

BYTE

the small systems journal





The 9-inch screen of the CT-VM monitor ($175) shown here with Southwest's new CT-64 illustrates the terminal's 64-character lines, switchable control character printing, and word highlighting. At just $500 for both, these matching units provide a complete CRT terminal with full cursor control, 110-1200 Baud serial interface, and many other features.

Now $325 buys a 64-character terminal kit

Our new CT-64 terminal kit gives you scrolling, full cursor control, 128-character ASCII display (with both upper and lower case), and two 1K memory pages. It's usable with any 8-bit computer.

Add our optional fully assembled 12 MHz CT-VM monitor for another $175 and you'll have the best CRT terminal buy offered anywhere.

The CT-64 gives you full cursor control, home-up and erase, erase to end of line or end of frame, cursor on/off, screen reversal, scroll or page, solid or blinking cursor, page selection, and end-of-page warning beeper.

The CT-64's features include:

- 64 or 32 characters per line (16 lines)
- Premium display with both upper and lower case letters, and descenders (g, j, etc.)
- Two 1K pages of 8-bit memory
- Scrolling or page mode operation
- 32 control character decoding
- Prints control characters (selectable)
- 128-character ASCII set
- 110/220 Volt 50-60 Hz power supply
- Highlights words with reversed background
- Optional 9-inch monitor with matching cover available
- Complete with keyboard, power supply, 110-1200 Baud serial interface, and case

Okay, Southwest, I know a bargain when I see it.

☐ Enclosed is $500 for the whole works (CT-64 terminal plus 12 MHz CT-VM monitor).
☐ Here's $325 for the CT-64.
☐ Send only data for now.
☐ Send me your $395 MP-68 computer kit.

☐ or BAC # ____________ Exp. Date ________

☐ or MC # ____________ Exp. Date ________

Name ____________ Address ____________

City ____________ State ________ ZIP ________

SWTPC

Southwest Technical Products Corp.
219 W. Rhapsody, San Antonio, Texas 78216

Circle 1 on inquiry card.

Meet the most powerful μC system available for dedicated work. Yet it's only $595.*

Here's the muscle you've been telling us you wanted: a powerful Cromemco microcomputer in a style and price range ideal for your dedicated computer jobs—ideal for industrial, business, instrumentation and similar applications.

It's the new Cromemco Z-2 Computer System. Here's some of what you get in the Z-2 for only $595:

- The industry's fastest μP board (Cromemco's highly regarded 4 MHz, 250-nanosecond cycle time board).
- The power and convenience of the well-known Z-80 μP.
- A power supply you won't believe (+8V @ 30A, +18V and −18V @ 15A — ample power for additional peripherals such as floppy disk drives).
- A full-length *shielded* motherboard with 21 card slots.
- Power-on-jump circuitry to begin automatic program execution when power is turned on.
- S-100 bus.
- Standard rack-mount style construction.
- All-metal chassis and dust case.
- 110- or 220-volt operation.

DEDICATED APPLICATIONS

The new Z-2 is specifically designed as a powerful but economical dedicated computer for systems work. Notice that the front panel is entirely free of controls or switches of any kind. That makes the Z-2 virtually tamper-proof. No accidental program changes or surprise memory erasures.

FASTEST, MOST POWERFUL μC

Cromemco's microcomputers are the fastest and most powerful available. They use the Z-80 microprocessor which is widely regarded as the standard of the future. So you're in the technical fore with the Z-2.

BROAD SOFTWARE/PERIPHERALS SUPPORT

Since the Z-2 uses the Z-80, your present 8080 software can be used with the Z-2. Also, Cromemco offers broad software support including a monitor, assembler, and a BASIC interpreter.

The Z-2 uses the S-100 bus which is supported by the peripherals of dozens of manufacturers. Naturally, all Cromemco peripherals such as our 7-channel A/D and D/A converter, our well-known BYTESAVER with its built-in PROM programmer, our color graphics interface, etc., will also plug into the S-100 bus.

LOW, LOW PRICE

You'll be impressed with the Z-2's low price, technical excellence and quality. So see it right away at your computer store—or order directly from the factory.

Z-2 COMPUTER SYSTEM KIT (MODEL Z-2K) (includes 4 MHz μP card, full-length 21-card-slot motherboard, power supply, one card socket and card-guide set, and front panel; for rack mounting) .... $595.

Z-2 COMPUTER SYSTEM ASSEMBLED (MODEL Z-2W) (includes the above as well as all 21 sockets and card guides and a cooling fan; for rack mounting) . . . $995.



Shown with optional bench cabinet

***kit price**

Cromemco
Incorporated
Specialists in computers and peripherals
2432 CHARLESTON RD., MOUNTAIN VIEW, CA 94043 • (415) 964-7400

In the Queue

JULY 1977 BYTE
Volume 2 Number 7

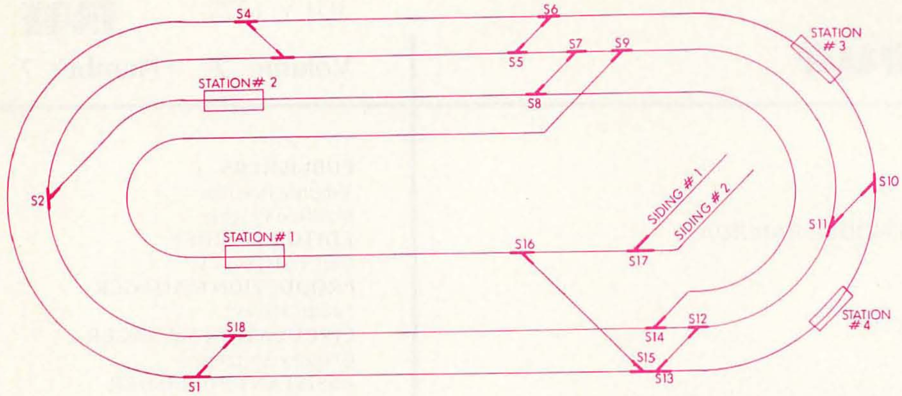
Foreground

Background

Nucleus

Cover: COMPUTER ENGINEERING by Robert Tinney

BYTE is published monthly by BYTE Publications Inc, 70 Main St, Peterborough NH 03458. Address all mail except subscriptions to above address; phone (603) 924-7217. Address all editorial correspondence to the editor at the above address. Unacceptable manuscripts will be returned if accompanied by sufficient first class postage. Not responsible for lost manuscripts or photos. Opinions expressed by the authors are not necessarily those of BYTE. Address all subscriptions, change of address, Form 3579, and fulfillment complaints to BYTE Subscriptions, PO Box 361, Arlington MA 02174; phone (617) 646-4329.
Second class postage paid at Peterborough NH 03458 and at additional mailing offices—USPS Publication No. 102410.
Subscriptions are $12 for one year, $22 for two years, and $32 for three years in the USA and its possessions. Add $5.50 per year for subscriptions to Canada and Mexico. $25 for a one year subscription by surface mail worldwide. Air delivery to selected areas at additional rates available upon request. $25 for a one year subscription by air delivery to Europe, or DM 60,— for a one year subscription through our European distributor in Darmstadt.
Single copy price is $1.50 in the USA and its possessions, $2 in Canada and Mexico, and $3 eleswhere, or DM 7,— through our European distributor in Darmstadt.
Foreign subscriptions and sales should be remitted in United States funds. Printed in United States of America.



page 44

In This BYTE

Model railroading can give the computer hobbyist an action packed computer application. Before tackling the job, however, it is important to have a firm understanding of what's involved in the design of a model railroad. In his article, **How to Computerize Your Model Railroad**, David C Brown explains in detail the problems faced by the model railroader and points to ways in which they can be solved. He then goes on to cover the requirements for microprocessor interfaces to the model railroad and some thoughts on the software of an operating system to give realism to the model.

Sometimes a small amount of hardware can speed up software considerably. A perfect example is provided by Tom Hall in his article which shows how **This Circuit Multiplies.** This circuit is a hardware multiplier which takes 8 bit operands and replies (ten clock periods later) with a 16 bit product. Here is an example of putting an inner loop into hardware, thereby speeding up an operation.

The model railroad is an ideal way for the personal computing enthusiast to enter the fascinating world of real time control: such a system is realistic but manageable. Authors Jack Hart and Ed Badger show you how they did it in **A Train Control Display Using the LSI-11 Microcomputer.**

Sometimes a bit of serendipity falls out of an application or project. Ken Barbier describes one such case in the form of his technique of using a television display circuit's ability to generate a raster with various synchronous patterns to debug and verify its operation. Read **The TV Oscilloscope.**

There are many ways to make a computer talk, but how do you get it to listen? **Speech Recognition for a Personal Computer System** discusses a topic which has fascinated and frustrated experimenters for years. Author James Boddie of Bell Labs (the pioneer researchers in the field) covers the history of the subject and presents a practical system which can be realized by a personal computing experimenter.

Real world systems may not be perfect, yet programming demands perfection. (If we drop a bit in one machine instruction, it becomes another instruction altogether.) As a background discussion of a large subject, W Douglas Maurer presents some information on **How to Pick up a Dropped Bit** using some of the techniques of error detection and correction.

What's a sure cure for memory megalomania? Why, **Give Your Micro a Megabyte** as outlined in Robert Grappel's short tutorial on some large memory design techniques which will prove adaptable to microcomputer systems. There should be virtually no reason to complain if his ideas were put into practice more universally.

An Introduction to Numbers, by Webb Simmons, serves as an introduction to the concepts of fixed, scaled and floating point numbers. Here you'll find some basic forms for each type, how the forms differ from each other, and how each can be used.

If you've ever been frustrated by the drudgery involved in relocating machine language programs with nothing but toggle switches, then Leor Zolman's **A Machine Code Relocator for the 8080** is for you! Just enter six key pieces of information and the program does the rest automatically, even fixing up all your address references.

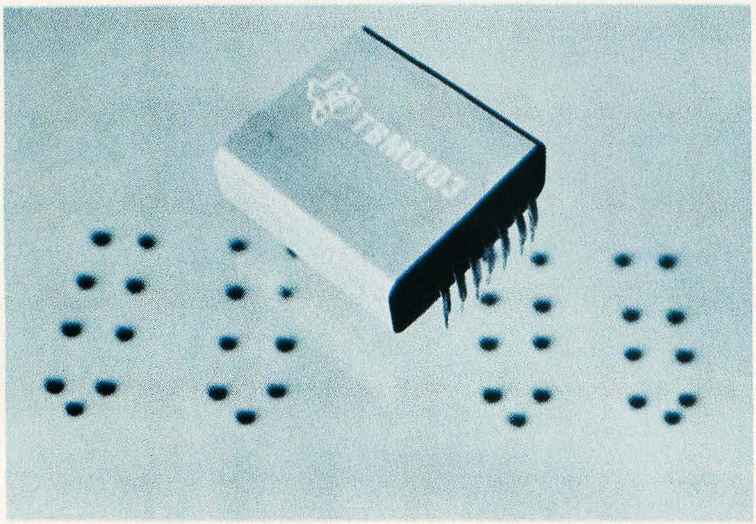
For beginners first learning about computers, we often get requests for some basic information. In **BASICally BASIC**, Robert Baker gives an informal introduction to the nature of the BASIC language and its uses in programming. Finding out what a typical BASIC can do is a good starting point in your personal evaluation of products available in the personal computing marketplace.

Given latitude and longitude of two points on the earth, how do you calculate the distance and bearing? If you use a little BASIC program by Rene Pittet, you can answer the question of **How Far – Which Way?** using a small processor.

If you have a Southwest Technical Products' TVT II, there is a simple circuit that you can add that will give you manual and computer control over the cursor's movement, erase and bell functions. In his article, **Add Cursor Control to Your TVT II,** Brother Thomas McGahee describes this simple circuit which can be attached to a TVT II.

Poor KIM. If one puts KIM-1 inside a fancy case, the built-in jewels of keyboard and displays get hidden. But Robert Grater comes to the rescue by **Giving KIM Some Fancy Jewels,** which consist of a remote set of displays connected to the processor by cable.

Editorial



This Elephant Never Forgets

By Carl Helmers

In a press release dated March 9 1977, Texas Instruments Inc made it official: a commercial bubble memory product is now available for general use, initially in sample quantities priced at $200 per chip. The device is the TBM0103, a single chip 92,304 bit storage element composed functionally of 144 shift registers with 641 bits per register. (Internally, according to TI, there are 157 registers, but the large size of the chip carries with it the attendant probabilities of defects, so discretionary manufacturing steps are used to connect 144 good registers per package.) The device has a complicated mechanical structure which includes two permanent magnet bias field elements, two orthogonal coils which modulate the bias field to produce a net rotating magnetic field in the garnet film bubble chip, and, of course, a magnetic shield to isolate the chip from external stray magnetic fields. The dynamic data characteristics of the chip are a bit transfer rate of 50,000 bits per second and an access time of 4 ms setup prior to transferring a 144 bit word of data into or out of the device. Conceptually, figure 1 illustrates what one of these memories looks like to a designer or programmer. This figure was constructed based on the limited information about overall characteristics in the press release. The procedure for writing a 144 bit word

Continued on page 58

NOW $2750. AMERICA'S LOWEST-PRICED PERSONAL COMPUTER SYSTEM WITH COLOR VECTOR GRAPHICS.



By taking advantage of the new technologies available to the industry today, we've consistently been able to give you one of the best prices on the market. Now because of great response, we can give you the best price. You can now buy the Compucolor 8001 for the reduced price of $2750. A complete stand-alone system with expanded graphics software for plotting points, vectors and bargraphs on a 160 x 192 addressable grid—in color. Eight independent background and foreground colors.

The Compucolor 8001 has an Intel 8080 CPU, 34 I/O ports and a color display with an effective band width of 75 MHZ compared to 5 MHZ for standard TV sets. In fact the Compucolor is the only totally integrated system on the market which includes a color display. You can also have special options for the Compucolor 8001 right now, including: Mini Disk Drives for extra memory, light pens and a variety of special keyboard features.

BASIC 8001 Is Easy To Learn. Compucolor's BASIC 8001 is a conversational programming language which uses English-type statements and familiar mathematical notations. It's simple to learn and easy to use, too. Especially when it comes to intricate manipulations or expressing problems more efficiently. The BASIC 8001 Interpreter runs in ROM memory and includes 26 statement types, 18 mathematical functions, 9 string functions and 7 command types for executing, loading, saving, erasing, continuing, clearing or listing the program currently in core.

Expandable Memory To 64K. The Compucolor 8001 has 11K bytes of non-destructible read-only memory which handles the CPU and CRT operating systems as well as BASIC 8001. Sockets are in place for an additional 21K of EPROM/MROM memory. The Random Access Main Memory has 8K bytes for screen refresh and scratch pad, 8K bytes for user workspace and room for 16K bytes of additional user workspace. The Compucolor also comes complete with a convenient mass storage device, Floppy Tape Memory. It's an 8-track continuous loop tape system, with a Baud rate of 4800 and an extra storage capacity of up to 1024K bytes per tape.

Color Graphics At Alphanumeric Black And White Prices.

That's what we're becoming famous for, and thanks to the tremendous response to the Compucolor 8001, we've been able to reduce our price even lower—to $2750. Look over our dealer listing on the adjacent page for the dealer nearest you. Then drop by for a demonstration. And while you're checking out the Compucolor 8001, check out your dealer's financing plan. He can help you turn a good deal into a good deal more.

Compucolor Corporation, P.O. Box 569, Norcross, Georgia 30091.

Compucolor® Corporation

Circle 4 on inquiry card.


Sol Terminal Computer
Processor Technology

Letters

ALTAIR (S-100) BUS QUESTION

I am using an 8080 based microprocessor system to control a video imaging system. A number of circuit boards are in use or being built which plug into the Altair style S-100 bus and work with video. Some of these boards require video sync signals (horizontal drive, vertical drive, composite blanking, composite sync and subcarrier). I would like to buffer these and put them on the bus as TTL signals. Could you please describe the known uses to date of the uncommitted pins of the Altair style S-100? I am looking for suggestions of a standardized place to put these five signals.

Carl Geiger
Onativia Crossing
LaFayette NY 13084

Can any readers supply an answer? One place to ask your question would be to call the engineering departments of various companies which make peripherals for the Altair bus.

ON PRINTERS

On April 1 1977, volume 2 number 4 arrived with the fine article by M S McNatt. I look forward to the next issue with the interfacing information. I truly believe that this is a most overlooked situation: the use of Baudot machines with the small computer. It would appear that many, many more people would be interested in the small computer if there was some method of retaining ASCII throughout the computer except in the use of the Baudot machine for input and output.

I must, however, take issue with the noise and smell of the Model 15. A Model 15 that is properly lubricated and adjusted, with its cover in place, should be something other than noisy; the cover should shield the owner from smell, also. If smoke comes out of the machine, something is wrong!

I note that Mr McNatt did not mention the Western Union strip printers; the model number does not come to mind, but I believe that they were 2-B. I saw an advertisement on this type of machine and I have reason to believe that the keyboard of such a machine will mate with the typing reperforator which usually (to my knowledge) comes without a keyboard.

Also, there is the Friden Flexowriter. My law office has a battery of those machines. The first one was acquired in 1958 and it clatters and bangs happily all day long. (Flexowriters are generally noisy, particularly the FL and SDS models.) Why is there not mention of

these machines? They are extremely sturdy and it is my understanding that one can still get maintenance for them. Many have input and output jacks or plugs.

Lucius B Dabney Jr
POB 947
Vicksburg MS 39180

A way of ducking the issue of your last question would be to point to the title of the article: "A Guide to Baudot Teletypes." As for oil and smoke, Mr McNatt's caricature of the Model 15 certainly fits my remembrances of using one on the amateur radio bands when I was in college . . . CH

DOES ANYONE HAVE THE LATEST ON 8080 FORTRAN?

Nine months ago, when I began to get serious about my own microcomputer, everybody was promising a FORTRAN compiler "just around the corner." A major manufacturer promised a FORTRAN compiler "soon" which degrades to "early 1977." Now they don't mention FORTRAN in their advertisements. As time goes on we hear less and less of a FORTRAN compiler. Here I sit with 40 K, access to timesharing and a pile of FORTRAN programs I can't use. I am a user, not a hardware man or a systems man. I would like to compile these and other FORTRAN programs one way or another and get on with using my machine.

FORTRAN may be a pterodactyl as Mr Lashley suggests, but it is a language commonly used to solve problems by those more interested in problem solving than in technique. Some of the hardware nuts would have us scrap everything everytime a new chip came along. In that event we would do nothing except construct new hardware. Mr Lashley would have us scrap years of training everytime a new programming idea appeared. That's OK for someone whose main interest is learning new languages. I don't feel we must scrap FORTRAN any more than Ma Bell has scrapped the old phone system. Sure, there have been problems, but by insisting the new must work compatibly with the old, she has built a good system for the user. The purists working for Bell who wish to scrap everything each time a new idea is discovered probably have ulcers.

So let us continue to build upon what we have and not scrap everything each time something new comes up. If this is the way things are done, I'll be updating my system far into the future. Now, I'd like to see some way of converting that pile of FORTRAN programs.

Lawson T Pierce
4891 Songbird Dr
Columbus OH 43229

LASHING LASHLEY

I nominate P M Lashley (Feburary 1977 BYTE, page 78) for the "Cybersnob of the Year" award for 1977. The year is young but I think he should get the nod; no one can possibly top his useless, self-serving, supercilious, unnecessary attack on another man's worthy efforts.

Roxton Baker
56 South Rd
Ellington CT 06029

APL INTEREST, AND CHARACTER GENERATION TUTORIAL NEEDED

I am writing in response to letters in the February 1977 issue of BYTE to say that I am building a microprocessor based APL capability. I plan to build an APL CRT terminal and will want a character generator chip. Since I plan to build my own but have been unsuccessful in finding a construction and theory article, BYTE could include such an article, or at least a bibliography, in a future issue.

Karl Gerhard
264 Taylor Rd
Portsmouth RI 02871

APL INFORMATION WANTED

Would you please send me some information concerning implementation of APL on microcomputers? Thank you for your consideration.

Gerry Wong
7950 Mercure
Brossard Quebec
CANADA J4Y 1A8

Watch future issues of BYTE.

COVER KUDOS

I had to write, though somewhat belatedly, to comment on the January 1977 cover painting by Robert Tinney. It was, I must admit, a rather emotional moment when I slid the wrapper off to reveal a painting which depicts exactly the feelings and aspirations I have held for some time now. The hope that the power of the computer, in the hands of all the right people, can somehow change the world from the smog filled grey and brown outside the window into the clear and beatiful vision seen through the terminal.

May I suggest that this cover be made available in poster size through BITS. While I did enjoy the Thomas Jefferson cover, I feel that we in the world of computing should be looking to the future, and that this particular painting

Continued on page 146

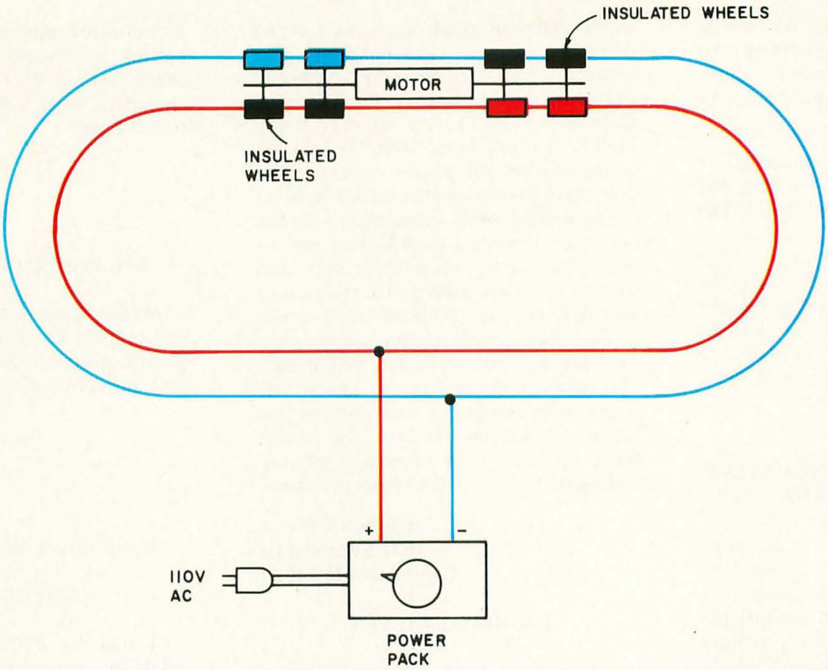


Figure 1: A simple train layout for one train showing the manner in which electricity is transmitted to the motor of the train.

How to Computerize

David C Brown
1704 Manor Rd
Havertown PA 19083

One of the many uses of computers, and especially personal computers, is in the area of real time process control operations. An enjoyable way of experimenting with this area would be to have your small computer control a model train layout. Those of us who attended the Personal Computing 76 convention held in Atlantic City NJ in August undoubtedly saw the Digital Equipment Corporation's display of a train layout controlled by an LSI-11 computer. There is no reason why you could not do something similar with your own system.

Since many of you may not be familiar with the techniques used in model railroading, let's start with a bit of background information. Figure 1 shows the simple oval layout most of us probably had as children in the US, where DC power has dominated the scene recently. Power is applied to each rail. One set of wheels on the engine picks up the positive side and feeds it to the motor; the other set of wheels picks up the negative. Unfortunately, this type of system can only handle one train running at a given time. If a second train were put on the tracks, we would not be able to control the two trains' speeds independently. To solve this problem, we go to block control, shown in figure 2. As you can see, the negative rail is common, as it was in figure 1. The difference lies in using insulators on the positive rails and two power packs. Now we can use switches so that either power pack can control either section of track. This allows us to have a train on the smaller loop (block 1) and another train on the run around track to the right (block 2), and we can control each train's speed independently through the two power packs. We still have to be a little careful when we operate this system, because if the train on block 2 runs onto block 1 while block 1 is occupied we will be right back where we started, with two trains running at the same speed and controlled by one power pack.

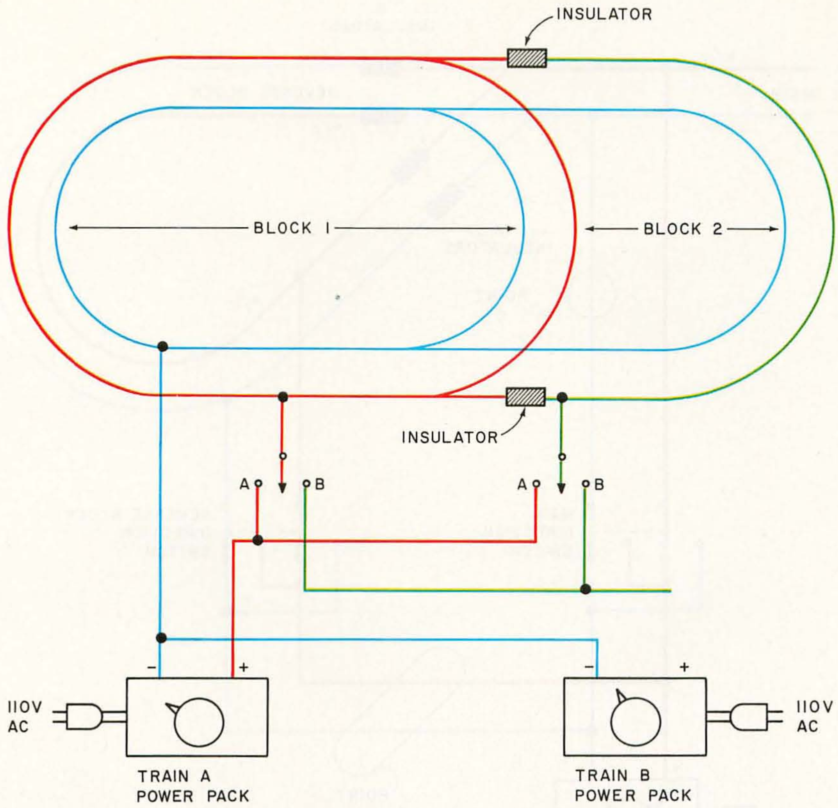


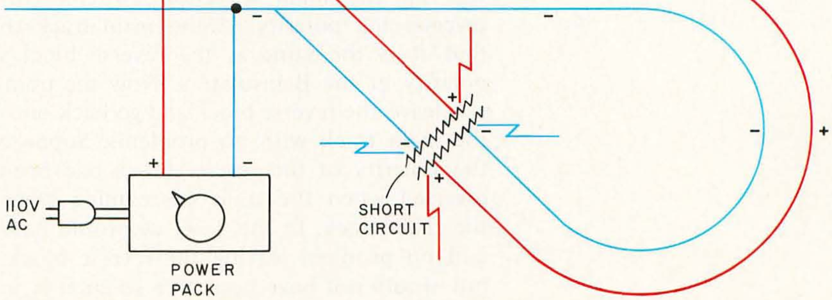
Figure 2: A basic example of block wiring. The negative rail is common to both blocks. The positive rail is divided by insulators into two sections. Each block can run a train using a separate power pack.

Your Model Railroad

In reality, then, two blocks are insufficient. A moderate size model railroad is likely to have from 12 to 24 blocks. Just before a train enters a block, the operator would flip that block's control to his power pack. When the train leaves a block, the operator would turn off that block so that the other operator could bring his or her train into the block if he or she wanted to.

The only other really important consideration in train layouts that we should discuss is the reverse loop. Suppose we had a loop as in figure 3, so that the train could go around the loop and end up on the same track but going the opposite direction. If you follow the polarities of the tracks around the loop, you will see that they end up as a short circuit, with "+" connected to "–" and vice versa. This condition requires the use of a special type of block, known as a reverse block or reverse loop. The reverse block is similar to the other blocks we have discussed, except that in this case, both tracks have insulators instead of just one. This is illustrated in figure 4. We have also added some other switches for reversing train directions. A normal power pack has a reverse switch that switches the polarity of its output to make the train run in the other direction. With reverse loops, we can no longer use this switch. Instead, we need a

Figure 3: An illustration of the type of difficulty that arises when a track is put through a reverse loop. Following polarities it will be seen that the negative side of the track will connect to the positive side at the end of the loop and vice versa.



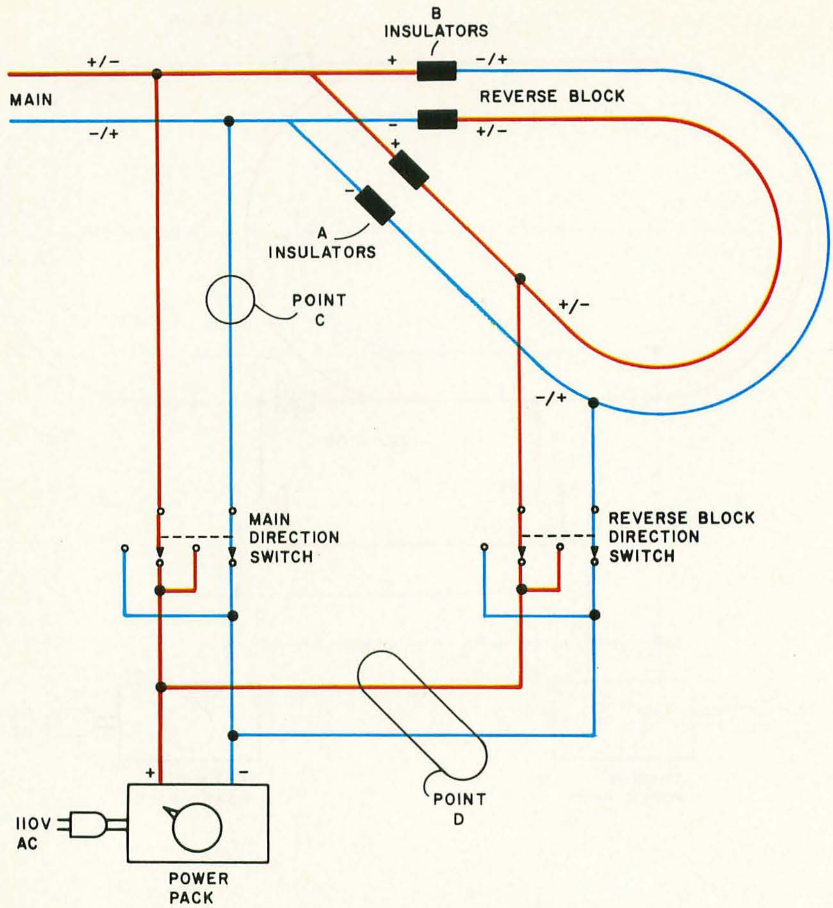


Figure 4: Using insulators on both the positive and negative tracks a block is created that is totally isolated from the main track. Using a separate power pack will allow the train to run through the block, turn around, and return to the main track.

direction switch for the reverse loop and another direction switch for the rest of the track. By playing with these switches, we can keep the polarities correct and prevent a short circuit. This is easier to see in an illustration. Assume that the train is coming down the main track from the left of figure 4 and that the track switch is positioned to the right so that the train will end up going around the reverse block in a counterclockwise direction. The polarities of the main and reverse block on each side of the A insulators are the same, so there is no problem crossing the insulators. If we leave everything the way it is though, we will have a problem when the train hits the B insulators, since the polarities are reversed on opposite sides of the insulators. In this case, while the train is on the reverse block, we flip the main direction switch. This reverses the polarity of the main track so that it is the same as the reverse block's polarity at the B insulators. Now the train can leave the reverse block and go back onto the main track with no problems. Suppose the polarity of the reverse block had been reversed when the train was coming down the main track. In this case, we would have had no problem leaving the reverse block, but would not have been able to enter it in the first place. The solution would be to flip the reverse block direction switch while the train was on the main track. This would match up the polarities at the A insulators. Then, after the train got onto the reverse block, we would reverse the main direction as we did before so that the train could safely leave the reverse block.

Two items were omitted from figure 4 for the sake of simplicity in the prior discussion. At point C we would tie in the second power pack and the block control switches of figure 2. At point D we would need a 2 pole 2 position switch to allow either power pack to control the reverse loop.

So much for train layouts and block control! The only other major piece of equipment on a model railroad is the track switch, called turnouts in model railroad terminology. Turnouts allow you to vary the route that your train takes, which is certainly more interesting than watching your train do nothing but go around in a circle. A turnout represents the actual track that can be switched. It physically can sit in one of two states which are electrically controlled. The electrical components for moving the turnout are called switch machines. Figure 5 shows a typical turnout and switch machine. The turnout contains a movable section. In

figure 5, the train will go straight. If the movable section were pulled down, the train would go to the left relative to its motion, up the turnout track. The switch machine is really just a pair of solenoid coils. A soft metal bar with an S shaped groove slides between the coils. The movable section of the turnout has a lever connected to it. The other end of this lever rides in the groove of the sliding bar of the switch machine. In figure 5, if the reverse switch was closed, coil B would be energized and the soft bar would slide to the right. The lever would ride in the groove and be pulled down, thus pulling down the movable section of the turnout. Since the coils draw close to 1 A when energized, they should only be energized long enough to move the turnout. If left energized for more than a few seconds, they could overheat and burn out. For this reason, push button switches are normally used for controlling the switch machines. The switch machines normally run off an alternating current accessory power pack and have no relation to the block system. They represent a separate, isolated set of circuits on the model railroad.

All of these electrical requirements may sound somewhat confusing. Linn Westcott has an excellent, very inexpensive paperback book entitled *How to Wire Your Model Railroad* from Kalmbach Publishing Company, 1959. This book, or one like it, should be available at any hobby shop. Another very good set of paperbacks is published by Atlas Tool Company, a leading manufacturer of model railroading equipment. Atlas has also simplified the wiring for you. They sell sealed switch units containing printed circuitry for all of the wiring I have discussed. The Atlas selectors contain block control circuitry, the Atlas controllers contain direction control and reverse loop circuitry, and the Atlas switch controllers contain the turnout control switches. These devices are inexpensive and have the ability to be plugged together for handling any size train system. All you do is connect your power packs, block wires and switch machine wires to the Atlas devices and your wiring is complete. The Atlas books also diagram track layouts, usually 6 to 8, from simple to complex, give construction details, and show how to wire them with the Atlas devices. What more could you ask for?

Now that I have you hooked, or at least interested in model railroading, let's take a look at what we need to control this system by computer. All of the principles discussed so far still apply. The only additional item that we need on the train layout itself is some way of knowing where the trains are.

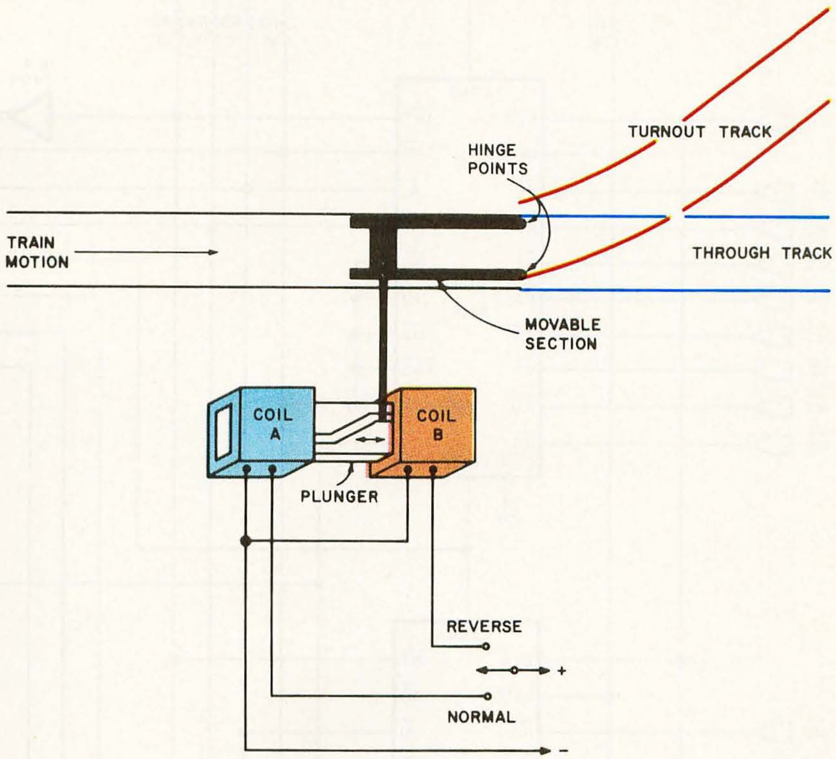


Figure 5: The basic workings of a turnout and switch machine. The two coils of wire move a piece of soft metal with a groove back and forth between them. The switch machine rides in the groove. When coil B is activated, by using the reverse mode, the metal is pulled into the B coil and the switch machine slides down in the groove changing the position of the switch machine.

The simplest approach would be to use magnetic reed switches. As shown in figure 6, these switches are nothing but two contacts, very close together, and sealed in a glass envelope. If we place these switches at intervals along the tracks and glue a small magnet to the bottom of our trains, the train going over the glass capsule will close the switch. These switches, by the way, are common in burglar alarm systems and other proximity sensing devices. I will refer to these switches along the tracks as markers in the remainder of this article.

The train control system to be discussed is currently under development. So far, the development has concentrated on the soft-

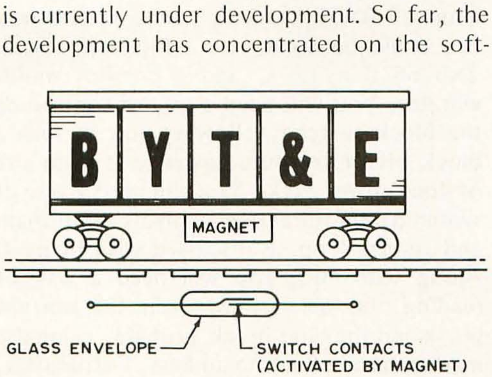


Figure 6: This is a magnetic reed switch which can be used as a marker along the track. The switch consists of two contacts that are enclosed within a glass envelope. The two metal contacts are activated magnetically when a car with a magnet attached passes over.

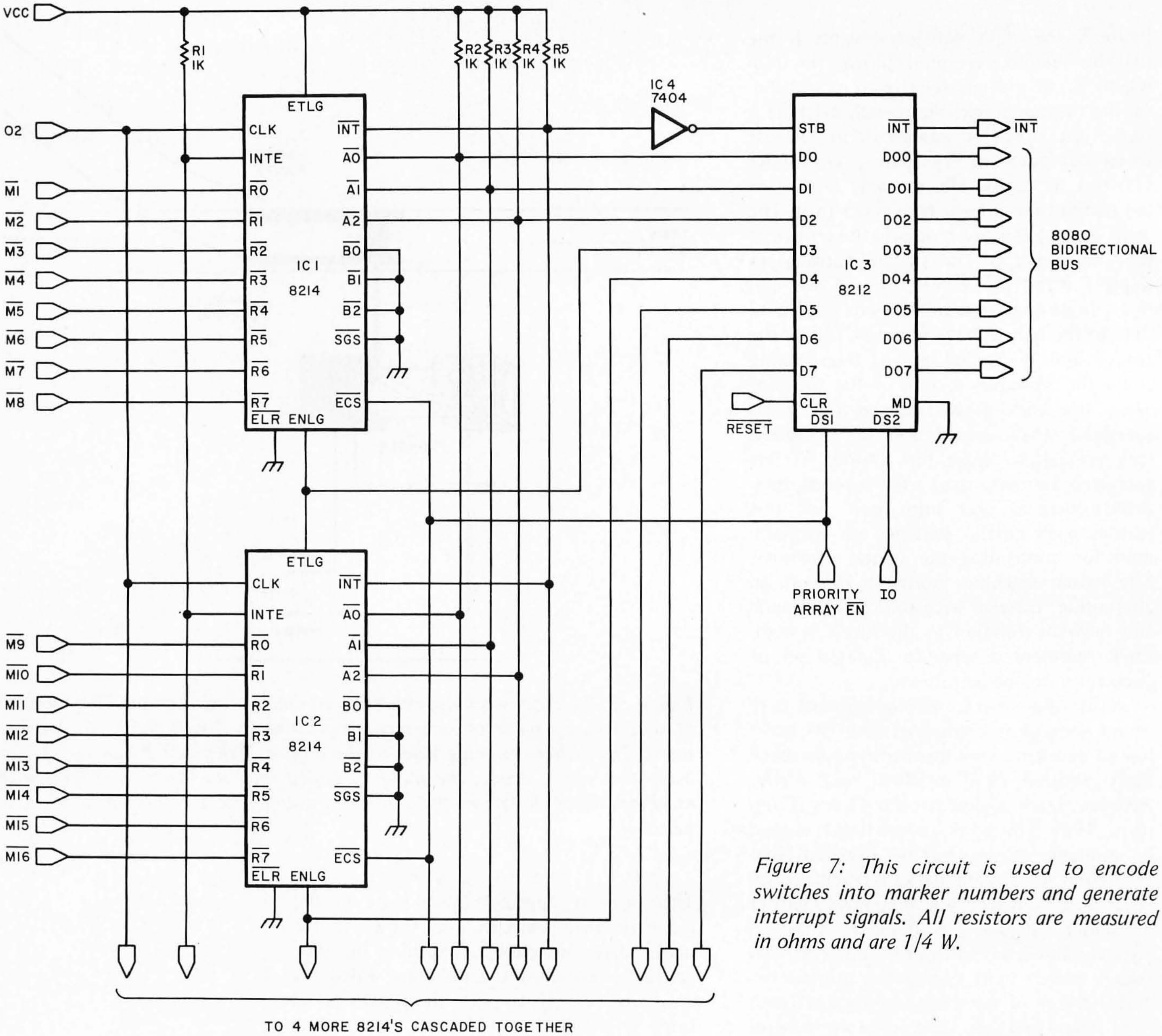


Figure 7: This circuit is used to encode switches into marker numbers and generate interrupt signals. All resistors are measured in ohms and are 1/4 W.

ware aspects, and most of the hardware details are yet to be worked out. These I must leave up to your ingenuity at the moment. You will need a way of controlling the switch machines. The October 1975 issue of BYTE, page 87, showed one way this could be done with a flip flop, several 2N3766 transistors, and a oneshot multivibrator. You will need a way of controlling the block switches, allowing you to turn a block off or to route power to it from any of your power packs. You will need a way of switching the direction controls, both main and reverse loop, as discussed with figure 4. Along with this, you will need a way of reading the polarity between the current block and the next block, probably using the marker number as the address. Fortunately, you only need this capability at entrances and exits from reverse blocks. You will need a way of controlling speed. This could be done with a digital to analog converter, or you could adapt the speed controller shown in the January 1975 issue of *Popular Electronics*, page 58. Last, but definitely not least, you will need the marker system.

The markers, as mentioned earlier, are the switches located along the track for sensing the location of the trains. When the train passes over a marker, an interrupt could be generated. The interrupt software must be able to read in a binary number identifying which marker was hit. This could be accomplished on an 8080 based system by cascading together Intel 8214 Priority Interrupt ICs, as shown in figure 7. Ideally, we would

like a large number of markers. Although I haven't the room to discuss this aspect, a graphics display could be used so that the track layout is displayed and each marker would light up as the train passed it. For this, we want many markers. The circuit of figure 7 will allow up to 40, but needs five 8214 chips, at $15 each. Another possibility would be to use a 256 word by 8 bit read only memory and have the markers control the address lines into the memory, which would be programmed with the marker numbers. I would be interested in hearing from anyone who has a cheap way of debouncing and encoding 256 switches into a byte of data that could be read in by an interrupt routine. For that matter, I would be interested in hearing from people on any of these hardware requirements, since I haven't worked them out myself yet.

So much for hardware. The software should actually be easier to explain than model railroading theory. The entire software package is table driven. This makes it easier to understand and easier to adapt to a wide variety of train layouts. The four major tables used are the marker table, the block table, the switch table and the train table. The marker table really controls most of the system. Each marker on the track has a corresponding entry in the marker table. When the train hits a marker, an interrupt is generated. Interrupts are probably preferable to a polled system, since it is crucial that none be missed and unless the marker event is latched, a busy processor could miss it. If your program is processing a console command and a train passes a marker before you get to poll it, you could have real chaos. The marker interrupt routine, then, would read in the marker number and look it up in the marker table. Table 1 shows the layout of an entry in the marker table. If a graphics display is used, MXY would be used to update the display, showing the train at the new marker location. MBLOCK would be the pointer to the block table entry for the block which contains this marker. MSETUP indicates that this is the last marker in the block. The software would go to the block table, figure out the next block that the train will enter, and issue a command to apply power to that block (remember figure 2?). MFREE tells the software that the marker is the first one in the block and power should be removed from the prior block. This is the reverse of MSETUP.

MFA and MFD are fractional accelerators and fractional decelerators. Fractionals represent a small change in the speed of the train. Any marker can be set up to add or subtract a fractional to the train's current speed. This would be used to speed up the train as it goes up a hill, called a grade, or slow it down on a downgrade. It could really be used any place you wanted the train's speed to automatically change. Fractionals can also be used for programmed stops. The MSTOP indicator in the marker table entry tells the system that the train should stop. However, a sudden stop would be unrealistic, so we can use the fractional decelerator. If the operator requests a stop at a particular stop number and that stop number equals MSTOPNO, the system will look at MSTOP and MFDS. If MFDS is on, the fractional decelerator will be applied, slowing down the train. MFDS also will prevent the fractional decelerator from being applied if a stop was not requested. Eventually the train will hit a marker that has the correct MSTOPNO and has the MSTOP indicator on, and the train will be stopped completely.

Each block on the track layout has a corresponding block table entry, as shown in table 2. The block table entry contains indicators for what train is in the block and whether the block is a reverse block (our figures 3 and 4 discussion). The reverse indicator will be used by the marker processing. When power is to be applied to the next

Label	Byte	Bit	Meaning
MMARKER	1 to 2	1 to 8	marker number
MBLOCK	3	1 to 8	block table entry displacement
MSTOPNO	4	1 to 8	stop number
MSETUP	5	1	setup next block indicator
MSTOP	5	2	stop indicator
MFDS	5	7	apply fractional decelerator on stop only
MFREE	5	8	release prior block indicator
MFA	6	1 to 4	fractional accelerator value
MFD	6	5 to 8	fractional decelerator value
MXY	7 to 8	1 to 8	XY coordinates for graphics display routine

Table 1: Marker Table Information. The purpose of the marker table is to organize the current information associated with each marker input from the model railroad layout. The list of information shown here is described in the text in more detail. The use of a graphic display of the layout is assumed by putting in display coordinate fields (MXY), which might be omitted otherwise. The basic marker entry information is repeated in this format for each marker.

Label	Byte	Bit	Meaning
BBLOCK	1	1 to 8	block number
BTRAIN	2	1 to 4	which train is currently in the block
BREV	2	8	reverse block indicator
BPRIOR	3	1 to 8	prior block ID
BNEXT	4	1 to 8	next block ID

Table 2: Block Table Information. The purpose of the block table is to organize the current information about each block in the model railroad layout. The block is identified by its number. (Although this field might be omitted if numbers and position in the table are made identical.) If the block currently has a train, its number is identified. The topology of the track network is reflected in the prior block and next block pointers.

Label	Byte	Bit	Meaning
SSWITCH	1	1 to 8	switch number
SPOS	2	1	through or turnout indicator
SFROM	3	1 to 8	from block ID
STHRU	4	1 to 8	through block ID
STURN	5	1 to 8	turnout block ID
SNMKTHRU	6	1 to 8	setup next marker ID for through block
SNMKTURN	7	1 to 8	setup next marker ID for turnout block
SPMKTHRU	8	1 to 8	release prior marker ID for through block
SPMKTURN	0	1 to 8	release prior marker ID for turnout block

Table 3: Switch Table Information. The switch table entries describe each switch in relation to the railroad layout by means of pointers to the "from," "through" and "turnout" blocks. This information, which is relatively fixed for a given layout, is used to modify the prior and next block information in the block table according to the settings of the switch. One such entry is needed for each switch in the layout. (A roundhouse would have to be a special case . . .)

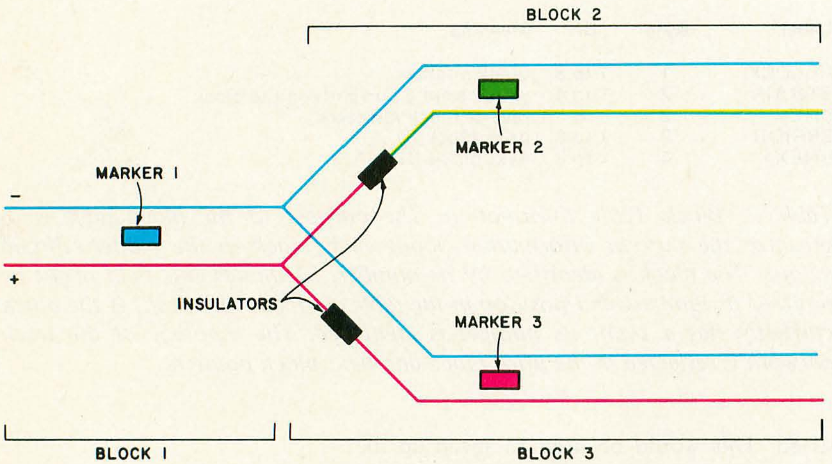


Figure 8a: An example of a marker being between two blocks. In this type of array the computer must only determine whether the train will go into block 2 or 3.

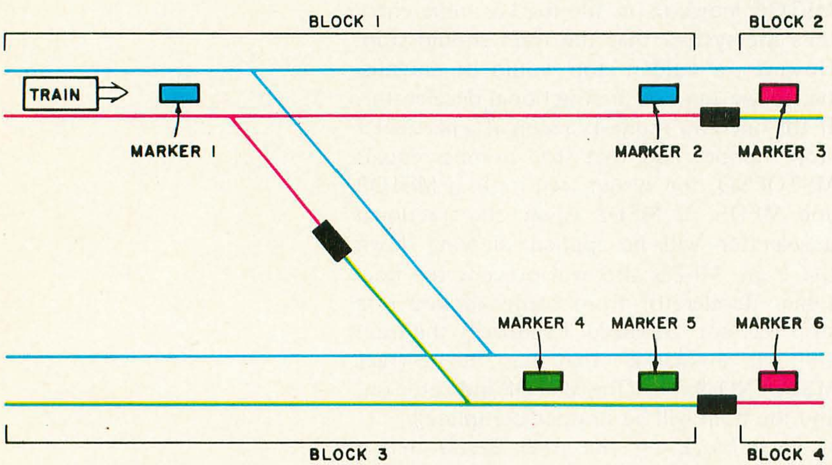


Figure 8b: This figure is slightly more complex than figure 8a. In this type of array the train may enter block 3 or block 2. The computer must determine which block the train will enter. If it is block 3 the information for block 3 must be updated for the particular train as soon as the train has passed marker 1. If the train will enter block 2 next, however, the information for block 2 must not be updated until the train has passed marker 2.

block, the software will check to see if either the current or the next block has the reverse block indicator on. If neither block is a reverse block, power is applied to the next block. However, if either block is a reverse block, we must do additional processing. Power is applied, but we have to worry about the polarity. Therefore, we read in the polarity between the two blocks, probably using the marker number as the address for the polarity interface. We cannot use the block number, since the polarity will be different at the two ends of the reverse block. If, after reading the polarity between the two blocks, we find that they are equal, we are done. If they are different and the current block is a reverse block, then we must reverse the polarity of the main track. If they are different and the current block is not a reverse block, we must reverse the polarity of the reverse block. These are the same operations that we discussed with figure 4, only the computer will do them rather than the operator. The BPRIOR and BNEXT fields in the block table entry are pointers to the block table entries for the prior and next block on the track layout. These pointers are maintained and modified by the switch processing.

When the operator requests that a particular switch position be changed, the software will locate the switch entry in the switch table. Table 3 shows the layout of a switch table entry. SPOS, the position indicator, will be toggled and the graphics display, if used, will be updated to show the new switch position. The new value of SPOS will be output to the switch interface. SFROM, STHRU, and STURN are used to update the prior and next block fields of the block table entries so that each block entry points to its correct successor and predecessor. SNMKTHRU, SNMKTURN, SPMKTHRU, and SPMKTURN are used to turn on and off the MSETUP and MFREE indicators in the marker table. For example, if the switch is set for turnout, MSETUP in the marker pointed to by SNMKTHRU will be turned off and will be turned on in the marker pointed to by SNMKTURN. SPMKTHRU and SPMKTURN do the same with the MFREE indicator in the marker table. The reason for this may not be apparent, so let's look at an example. In figure 8a, this processing would not be needed. MSETUP in marker 1 would always be on, and MFREE could be left on in both marker 2 and marker 3 since there is no other way into block 2 or 3 other than from block 1. However, figure 8b is a different case. Here we have the switch in the middle of a block. If the switch were set for the

turnout, the train would leave block 1 and enter block 3. Marker 1 would have MSETUP on and marker 4 would have MFREE on. Now let's change the switch position. The train will no longer go into block 3. The switch processing would have updated block 1's block table entry so that block 2 was the next block. If we try to set up the next block (block 2) according to marker 1, we will be premature. Maybe there is another train in block 2 at the moment, since there could be quite a bit of track between the switch and the beginning of block 2. Likewise, if a train comes down block 3 and tries to free the prior block according to marker 4, we might have a similar problem. The switch processing, then, must turn off MSETUP in marker 1 and turn off MFREE in marker 4 if the switch is set for through and turn them on if the switch is set for turnout. In this particular case, markers 2 and 5 could have MSETUP on all the time, markers 3 and 6 could have MFREE on all the time, SNMKTHRU and SPMKTHRU will be zero, and SNMKTURN and SPMKTURN will have a 1 and a 4 respectively.

The train table of table 4 has one entry for each train that you are running. This table entry is used primarily for keeping track, pardon the pun, of where each train is. It specifies what block the train is in, the last marker hit, the train's direction (forward or reverse), the train's current speed, and the port address for the train's power pack interface. This interface would control the speed, through digital to analog convertors or similar, and the main and reverse block direction controls. The train table entry would be updated by marker processing and by speed and direction commands issued by the operator.

So far, we have discussed primarily the automatic functions of the software. Now let's look at the functions controlled by the operator. We already discussed switches and stops. That leaves us with direction and speed. Suppose we want to reverse a train. First, this command would not be allowed if the train was moving, since we might cause a derailment if we suddenly reversed a fast moving train. Other than that, it's fairly simple. We store an indicator in TDIR of the train table entry to indicate that the train is going in the reverse direction and then issue the command to reverse either the main polarity or the reverse block polarity, depending on whether or not the train is currently in a reverse block. From then on, we reverse logically the meaning of the MSETUP and MFREE indicators in the marker table entries and we reverse logically

Label	Byte	Bit	Meaning
TTRAIN	1	1 to 4	train number
TDIR	1	8	direction
TBLOCK	2	1 to 8	current block ID
TMARKER	3 to 4	1 to 8	current marker ID
TSPEED	5	1 to 8	train's current speed
TPORT	6	1 to 8	IO port address for controller

Table 4: Train Table Information. Each train in the layout must also be followed in the software as it runs its route. The information can be formatted and referenced with a structure such as this. The details vary for each train with time, as the train progresses through the layout.

the meanings of the prior and next block pointers in the block table entries. Therefore, we set up the next block when MFREE is on and free the prior block when MSETUP is on. BNEXT points to the prior block and BPRIOR points to the next block.

Train speed control can be somewhat more difficult. There are basically two ways to control speed. The easiest would be to use a digital to analog converter. Trains run on 0 to 16 VDC, so a converter would be needed capable of providing this range at approximately 1 A per train to be run at a time. Therefore, if you plan on running two trains at the same time, you need a 0 to 16 VDC supply rated at 2 A minimum. This idea works well with the fractional system outlined earlier, but is not very realistic. The operator would have to enter on the console the voltage to be applied and the software would output to the converter the correct value. To gradually speed up a train would keep the operator busy for a while entering speed commands.

A more realistic approach would be to enter an accelerate command that would gradually speed up the train at a uniform rate. When the proper speed is reached, the operator would enter a hold command to maintain the current speed. To slow down, a decelerate command would cause the train to gradually slow down, and the hold command would again be used to stop decelerating. This type of control could be accomplished by adapting the circuit on page 58 of the January 1975 *Popular Electronics* to operate as an IO device rather than by push buttons. This complicates the fractional system, since the fractionals were intended to represent actual voltage quantities to be applied to the train's speed.

A similar approach could be to still use digital to analog converters, but to have the software periodically increment or decrement the value applied to the converter if an accelerate or decelerate command was in effect. This method will satisfy both realism and the operation of the fractionals, but depends upon the construction of the soft-

M

Marker Interrupt Processing Routine

The M algorithm is used for marker interrupt processing. It reads in the marker number, performs operations on it, including block processing (B algorithm) if necessary, determines any change in speed and then returns from the interrupt.

M1. Read the marker number input and look it up in Marker Table.
M2. Use MBLOCK of Marker Table entry to point to Block Table entry.
M3. Use BTRAIN of Block Table entry to point to Train Table entry.
M4. Store marker and Block Table pointers in TMARKER and TBLOCK.
M5. If TDIR=0 and MSETUP is on then go to B1.
M6. If TDIR=1 and MFREE is on then go to B1.
M7. If TDIR=0 and MFREE is on then go to B11.
M8. If TDIR=1 and MSETUP is on then go to B11.
M9. Send MXY to graphics display if it is used.
M10. If MFA > 0 then add MFA to TSPEED and send TSPEED output to the speed controller.
M11. If MFD > 0 and MFDS is off then subtract MFD from TSPEED and send TSPEED output to the speed controller.
M12. If MFD > 0 and MFDS is on and stop has been requested at MSTOPNO then subtract MFD from TSPEED and send TSPEED output to the speed controller.
M13. If MSTOP is on and a stop has been requested at MSTOPNO then store 0 in TSPEED and send TSPEED output to the speed controller.
M14. Return from interrupt processing routine.

B

Block Processing Routine

The B algorithm is used for block processing of the power controls. The algorithm determines the voltage and polarity changes that are to be made for the various blocks. Lines B1 to B10 set up the next block for the train to enter it. Lines B11 to B15 free the prior block that the train is leaving.

B1. If TDIR=1 then use BPRIOR as the next block pointer, else use BNEXT as the next block pointer.
B2. If BTRAIN of the next Block Table entry = TTRAIN then go to M7.
B3. If BTRAIN ≠ 0 then stop system (there is a train over run: two trains in same block!)
B4. Store TTRAIN in BTRAIN of the next block.
B5. Send command output to the block controller to power up the next block.
B6. If BREV of the current block and BREV of the next block are both 0 then go to M7.
B7. Read polarity of tracks using MMARKER of the current Marker Table entry as the address.
B8. If polarity=0, then current and next block polarities are the same so go to M7.
B9. If BREV of the current block = 1, then output a command to reverse the main polarity and go to M7.
B10. Output a command to reverse the reverse block's polarity and go to M7.
B11. If TDIR=1 then use BNEXT as the prior block pointer else use BPRIOR as the prior block pointer.
B12. If BTRAIN of the prior block = 0 then go to M9.
B13. Store 0 in BTRAIN of the prior block.
B14. Send command output to block controller to turn off prior block and go to M9.

S

Switch Command Routine

The S algorithm performs the switch command processing by resetting all of the markers that are being affected by a particular train. It is called by an operator command.

S1. Look up the switch number in the Switch Table.
S2. Reverse value of SPOS (0=through, 1=turnout).
S3. If SPOS=0 then use STHRU as the pointer to next block and go to S5.
S4. Use STURN as the pointer to the next block.
S5. Store the pointer to the next block in BNEXT of the Block Table entry pointed to by SFROM.
S6. If SPOS=1 then go to S14.
S7. Turn off MSETUP in Marker Table entry pointed to by SNMKTURN.
S8. Turn on MSETUP in Marker Table entry pointed to by SNMKTHRU.
S9. Turn off MFREE in Marker Table entry pointed to by SPMKTURN.
S10. Turn on MFREE in Marker Table entry pointed to by SPMKTHRU.
S11. Send SPOS output to the switch controller using SSWITCH as address.
S12. If the graphics display is used then update it to show new switch position.
S13. Return to operator command processor.
S14. Turn off MSETUP in the Marker Table entry pointed to by SNMKTHRU.
S15. Turn on MSETUP in the Marker Table entry pointed to by SNMKTURN.
S16. Turn off MFREE in the Marker Table entry pointed to by SPMKTHRU.
S17. Turn on MFREE in the Marker Table entry pointed to by SPMKTURN.
S18. Go to S11.

R

Reverse Direction Command Routine

The R algorithm performs the reverse direction command processing. If the train is not moving, the polarity of the block will be reversed and TDIR will be logically inverted.

R1. If TSPEED in the current Train Table entry ≠ 0 then indicate invalid request and return to command processor.
R2. Reverse the present value of TDIR in current Train Table entry.
R3. Use TBLOCK of the current Train Table entry to point to the current Block Table entry.
R4. If BREV of the current Block Table entry = 1 then go to R7.
R5. Send command output to reverse the polarity of the main track.
R6. Return to operator processor.
R7. Send command output to reverse polarity of reverse block.
R8. Return to operator processor.

ware to increment or decrement at a reasonable pace.

Table 5 shows a list of operator commands that might be used on a typical computerized model railroad. The remainder of this article is devoted to some algorithms for the more important processes we have discussed. Algorithms, if the term is unfamiliar, are simply flowcharts expressed in words rather than in blocks and arrows. Since this article is not presenting actual code and is more directed to theory than to bit by bit details, generalized algorithms seem more appropriate. I hope this article has increased your interest in the area of model railroading and has given you some ideas on how you can simulate, on a small scale, the types of real time process control systems that could be used on a large scale to control an actual railroad.■

Mnemonic	Command meaning
A Tx	accelerate train x
D Tx	decelerate train x
H Tx	hold train x at current speed
S Tx	stop train x immediately
S Tx Sy	stop train x at stop number y
FA xxx My	set fractional accelerator value of xxx into marker y
FD xxx My	set fractional decelerator value of xxx into marker y
FD xxx My,S	same as above but also turn on MFDS
R Tx	reverse train x
Sx	toggle position of switch x
Ex My1,y2,--,yn	establish stop x at markers y1,y2,--yn and turn on MSTOP in marker yn
Dx	delete stop x
K	kill system; emergency power off
C	close down system; orderly stop of trains and power off

Table 5: Definition of a Command Language. Here is mnemonic set of commands for control of the trains on a model railroad layout from a terminal keyboard. When a general purpose keyboard is used, this sort of sequence is required – although the particular commands defined here are not necessarily the only set possible. Algorithms S and R (see separate boxes) are command processing routines for switch toggling and reversal of trains, two of the more complex commands in this list.

GLOSSARY

Algorithm: A method used to express the steps to be taken in solving a problem. It may be expressed in words, as a flowchart of blocks, lines and arrows, or in other forms.

Block control: Sectioning up a railroad layout so that multiple power packs can be used to run more than one train on the tracks at a time.

Block table: A table used in the software for controlling a model train layout. Each block on the tracks has a corresponding entry in the block table.

Digital to analog converter: Commonly abbreviated as DAC, the device takes as input a digital value, typically an "n" bit binary number, and outputs an analog signal in the form of a variable voltage corresponding to the digital input value.

Fractionals: As used in this article, a fractional is a 4 bit binary number representing a small change in the speed of the train. It is used for automatic speed changes under program control.

Graphics display: A video display device that can display lines, curves and drawings rather than just characters. In a train system, a graphics display could be used to show a real time diagram of the train layout and where each train is located.

Insulators: In a train layout, insulators are used to isolate sections of track in order to use block control. Since power is supplied to the train through the tracks, insulators are placed between sections of track so that each section can be controlled independently.

Markers: Switches located on the track that are activated by the passing of a train. These markers are used by the software to keep track of where the trains are.

Marker table: A table used in the software, containing an entry for each marker on the train layout.

Polarity: As used in this article, polarity refers to which track is positive and which track is negative. Normally, this is not really important, except in the case of reverse loops, where the polarity between two adjacent blocks on the layout might be different, resulting in a short circuit when crossed by the train.

Reverse loop: Or reverse block; this is where a section of a track will allow a train to go around in a loop and end up on the same track but going in the opposite direction. Special handling of these cases is needed.

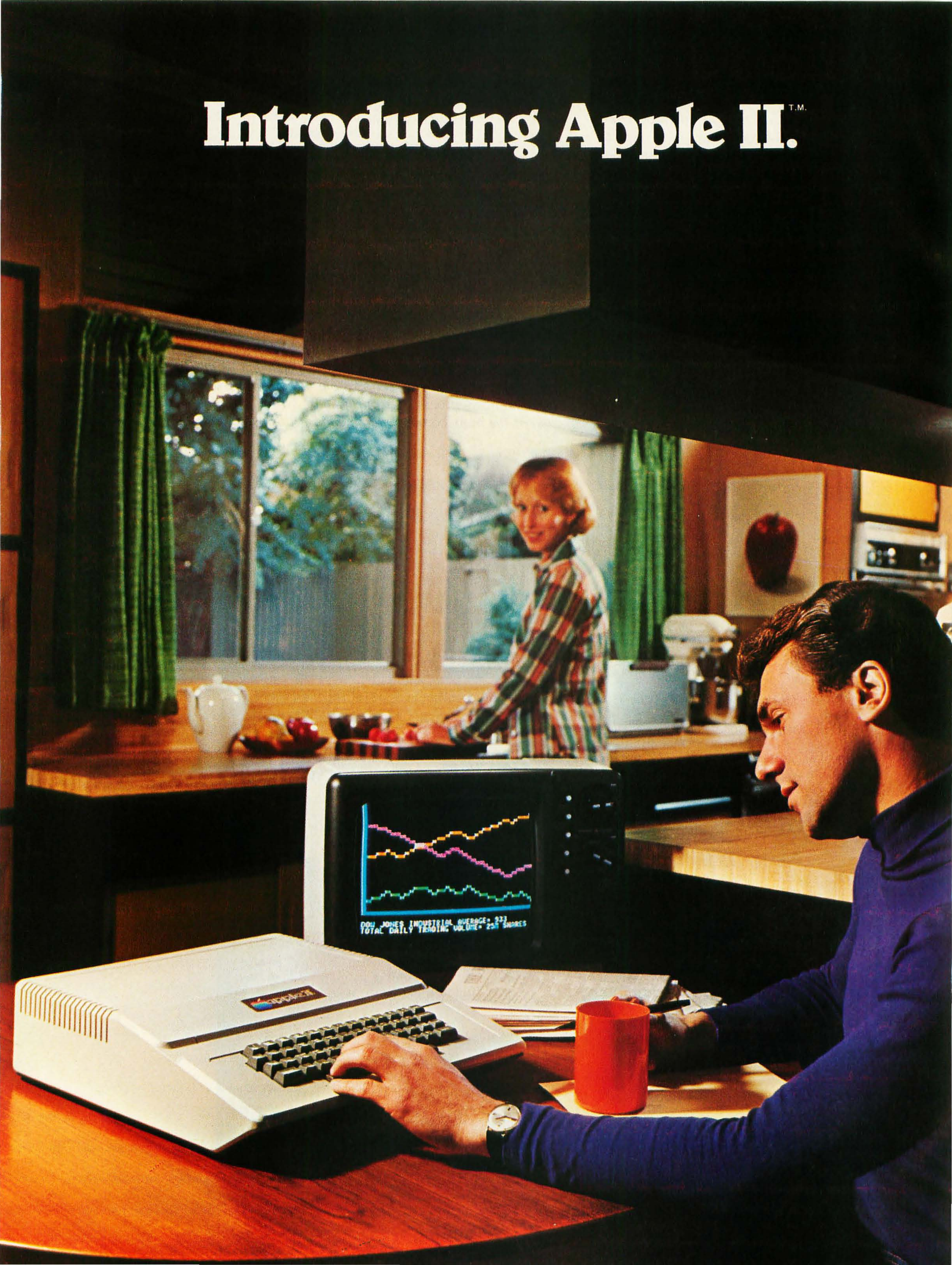
Solenoid: A coil of wire which acts as an electromagnet. When current is applied to the coil, a bar of soft metal will be attracted toward the magnet. In model railroading, solenoids are part of a switch machine.

Switch machine: A solenoid activated mechanism attached to a turnout, used to move a section of track so that the route of the train can be modified.

Switch table: A table used in the software, containing an entry for each switch machine and attached turnout on the track.

Train table: A table containing an entry for each train that is running on the train layout (also, the physical structure on which your train layout is built).

Turnout: A movable section of track, used to modify the route of the train. A turnout allows the train to go in one of two possible directions. A switch machine is used to move the turnout. A turnout has a normal position, which is the direction most commonly taken, and a reverse position, which is the less frequently used position. In this article, I have also used turnout to refer to the reverse position, and through to refer to the normal position.


Introducing Apple II.™

You've just run out of excuses for not owning a personal computer.

Clear the kitchen table. Bring in the color TV. Plug in your new Apple II,* and connect any standard cassette recorder/player. Now you're ready for an evening of discovery in the new world of personal computers.

Only Apple II makes it that easy. It's a



complete, ready to use computer, not a kit. At $1298, it includes video graphics in 15 colors. It includes 8K bytes ROM and 4K bytes RAM—easily expandable to 48K bytes using 16K RAMs (see box). But you don't even need to know a RAM from a ROM to use and enjoy Apple II. For example, it's the first personal computer with a fast version of BASIC permanently stored in ROM. That means you can begin writing your own programs the first evening, even if you've had no previous computer experience.

The familiar typewriter-style keyboard makes it easy to enter your instructions. And your programs can be stored on—and retrieved from—audio cassettes, using the built-in cassette interface, so you can swap with other Apple II users.

You can create dazzling color displays using the unique color graphics commands in Apple BASIC. Write simple programs to display beautiful kaleidoscopic designs. Or invent your own games. Games like PONG—using the game paddles supplied. You can even add the dimension of sound through Apple II's built-in speaker.

But Apple II is more than an advanced, infinitely flexible game machine. Use it to teach your children arithmetic, or spelling for instance. Apple II makes learning fun. Apple II can also manage household finances, chart the stock market or index recipes, record collections, even control your home environment.

Right now, we're finalizing a peripheral board that will slide into one of the eight available motherboard slots and enable you to compose music electronically. And there will be other peripherals announced soon to allow your Apple II to talk with another Apple II, or to interface to a printer or teletype.



Apple II is designed to grow with you as your skill and experience with computers grows. It is the state of the art in personal computing today, and compatible upgrades and peripherals will keep Apple II in the forefront for years to come.

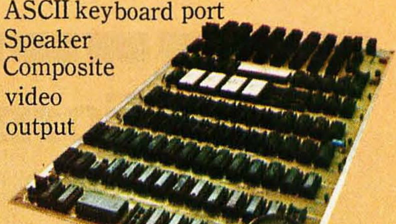
Write us today for our detailed brochure and order form. Or call us for the name and address of the Apple II dealer nearest you. (408) 996-1010. Apple Computer Inc., 20863 Stevens Creek Boulevard, Bldg. B3-C, Cupertino, California 95014.



Apple II™ is a completely self-contained computer system with BASIC in ROM, color graphics, ASCII keyboard, lightweight, efficient switching power supply and molded case. It is supplied with BASIC in ROM, up to 48K bytes of RAM, and with cassette tape, video and game I/O interfaces built-in. Also included are two game paddles and a demonstration cassette.

SPECIFICATIONS

- **Microprocessor:** 6502 (1 MHz).
- **Video Display:** Memory mapped, 5 modes—all Software-selectable:
  - Text—40 characters/line, 24 lines upper case.
  - Color graphics—40h x 48v, 15 colors
  - High-resolution graphics—280h x 192v; black, white, violet, green (12K RAM minimum required)
  - Both graphics modes can be selected to include 4 lines of text at the bottom of the display area.
  - Completely transparent memory access. All color generation done digitally.
- **Memory:** up to 48K bytes on-board RAM (4K supplied)
  - Uses either 4K or new 16K dynamic memory chips
  - Up to 12K ROM (8K supplied)
- **Software**
  - Fast extended BASIC in ROM with color graphics commands
  - Extensive monitor in ROM
- **I/O**
  - 1500 bps cassette interface
  - 8-slot motherboard
  - Apple game I/O connector
  - ASCII keyboard port
  - Speaker
  - Composite video output



Apple II is also available in board-only form for the do-it-yourself hobbyist. Has all of the features of the Apple II system, but does not include case, keyboard, power supply or game paddles. $598.

PONG is a trademark of Atari Inc.

*Apple II plugs into any standard TV using an inexpensive modulator (not supplied).

apple computer inc.™

Order your Apple II now.

Use this order form to get your Apple II fast. As a special offer for those who order now, we will include free a custom vinyl carrying case (a $50 value). And we will also pay shipping charges to anywhere in the continental United States.

Apple II Price List.

RAM Complement	Apple II System	Calif. Residents Add	Apple II Board-only	Calif. Residents Add
4K	$1,298.00	$ 84.37	$ 598.00	$ 38.87
8K	1,398.00	90.87	698.00	45.37
12K	1,498.00	97.37	798.00	51.87
16K	1,698.00	110.37	978.00	63.57
20K	1,778.00	115.57	1,078.00	70.07
24K	1,878.00	122.07	1,178.00	76.57
32K	2,158.00	140.27	1,458.00	94.77
36K	2,258.00	146.77	1,558.00	101.27
48K	2,638.00	171.47	1,938.00	125.97

Memory is offered at a 20% savings when ordered with the system–or board–as reflected in the prices above.

Additional RAM can be easily added-in at a later date as your needs develop.
One set 4K chips (4K bytes) $125
One set 16K chips (16K bytes) $600

Prices and specifications subject to change without notice.

apple computer inc.™
20863 Stevens Creek Blvd., B3-C
Cupertino, California 95014
(408) 996-1010

Order Form

☐ Please send me an Apple II System
☐ Board Only
with ____ K bytes of RAM (4K minimum) at $________
California Residents add 6.5% tax ________
Total $________
Name ________
Address ________
City ________
State ________ Zip ________
Phone ________
Shipping Address (if different)
____
____

☐ Cashier's check or money order enclosed.
(Please allow 2 additional weeks for personal checks.)
Please charge to my
☐ BankAmericard
☐ VISA
☐ Master Charge
Card Number ________
Expiration Date ________
Signature ________

Mail to: **Apple Computer Inc., 20863 Stevens Creek Blvd., B3-C, Cupertino, California 95014**




Photo 1: Two different shots from the balcony of the Civic Auditorium show the extent of the crowd. At the end of the show, chaircreature Jim Warren announced that 12,800 people had attended.

Random Observations and Conversations at the First West Coast Computer Faire

Notes and Photos by Lawrence F Willard

The kids monopolized the computer games most of the time, and were better in their scoring than most of the adults. Said one man to another:

"Those kids have been at the keyboard for 15 minutes; shall we push 'em out of the way?"

"No," said the other. "They're doing a helluva lot better than I know I could do."

There were several variations of Star Trek in evidence. At the Processor Technology booth one frustrated kid kept getting this printout whenever he entered a command:

"Captain, I'm afraid your last command made no sense."

He was lucky. A kid about 12 years old playing Star Trek at the Vector Graphic Inc booth apparently made a king sized error and got this printout:

"Thanks to your bungling the galaxy has been destroyed by the Klingon War Fleet. You are demoted to cabin boy. Try again."

The boy turned around and saw me watching him.

"Don't say anything," he growled.

These games reveal a lot about people to dedicated people watchers like me. At the National Semiconductor booth a computer was set up to plot, on a screen, a graph of the player's physical, emotional and intellectual cycles. You were asked to key in your birth date and to indicate over how many days you wanted the chart plotted. I noted, curiously, that some people used arbitrary dates or somebody else's birth date rather than their own. Are some people reluctant

Larry Willard, who writes regularly for Yankee *magazine (Dublin NH), acted as BYTE's roving eye at the First West Coast Computer Faire held in San Francisco's Civic Auditorium on April 16 and 17 of this year. Here are Larry's notes on Random Observations and Conversations at the event.*



Photo 2: The mild, unassuming roll top desk at the left of this sequence hides an intelligence. For, when all the doors and slides of this "desk" by RDC Enterprises (8352 Stanford Av, Garden Grove CA) are opened, they reveal an experimenter-oriented computer with built-in Altair bus extender card at desk level, a keyboard at desk level and video monitor. The actual computer mainframe is below desk level behind doors, and the bottom drawer can hold the experimenter's tools.



Rich Man, Poor Man, Merchant, Physician, Teacher, Lawyer, Student, Musician....



There's an iCOM Floppy Disk System for Everyone!

More Uses

People from every walk of life are adding iCOM® Floppy Disks to their microcomputers for such diverse tasks as payroll, inventory control, mailing lists, game playing, record keeping, parts ordering . . .

We've uncovered some innovative applications, too: The sailboat architect who puts equations and algorithms on an iCOM disk to test his nautical theories; the student who has automated a bowling alley; the iCOM dealer who designed an environmental control system for a university.

More Speed

These users have found iCOM floppies to be much faster and more versatile than cassette or paper tape. With iCOM, programs can be loaded in seconds; files updated in minutes; hundreds of programs can be stored on a single disk.



More Models

iCOM has Frugal Floppies™, Dual Floppies, Microfloppies™ (using the new 5¼" diskette), and other new approaches to floppy disk systems. Each is hardware and software compatible with Altair™, IMSAI, Poly 88, Sol-20 and other microcomputers using the Altair S-100 bus format.



More Software

Then there's iCOM's famous software: Powerful field-proven FDOS-II with macro-assembler, string-oriented text editor, and file manager. Plus easy-to-use compatible 8K Disk BASIC. Each with super features such as: named variable length files, auto-file create, open and close, multiple merge and delete . . . and more.

More Backup

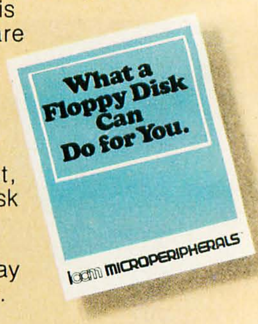
We've been building floppies for microcomputers for more than 3 years. Long before the rest. Thousands of systems are operating perfectly in the field. And we're part of Pertec Computer Corporation, one of the largest manufacturers of peripherals, microsystems, data entry products and data processing systems. We'll be around whenever you need us.

More Dealers

Maybe not in quantity, but in quality. We've chosen our dealer network carefully to assure you of assistance every step of the way. Our prices are right. Our delivery is fast. Our dealers are experienced and knowledgeable.

Must Reading

Our free booklet, "What a Floppy Disk Can Do for You" is must reading. Send for yours today or visit your dealer.



icom® MICROPERIPHERALS®

a division of Pertec Computer Corporation

6741 Variel Ave., Canoga Park, CA 91303 U.S.A.
Tel. (213) 348-1391 TWX 910-494-2788

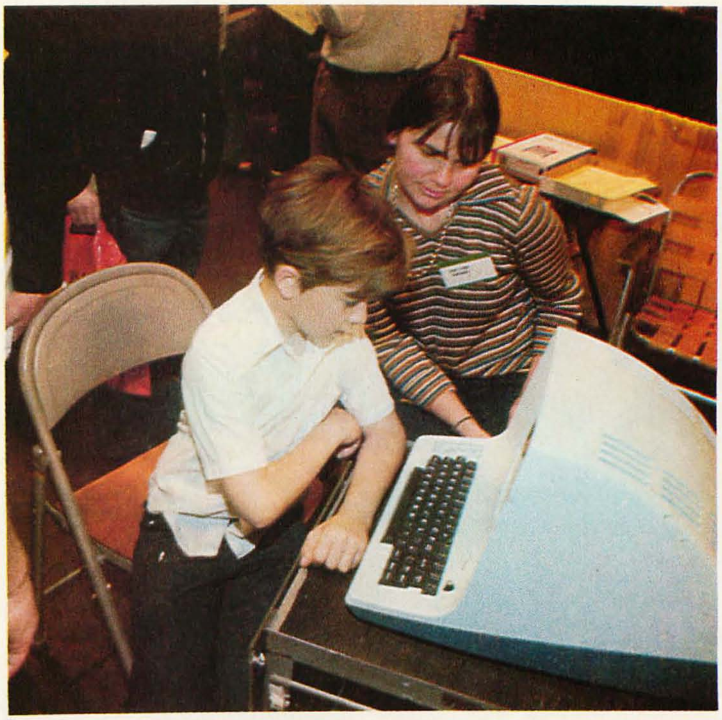


Photo 3: The kid at the keyboard of this IMSAI 8080 under the watchful eye of Lynn Stone is Bob Bowdige, a fifth grader at Rollingwood School in San Bruno. At this school, 200 students each week study BASIC programming. The school was invited to become part of the exhibit at the Action Audio Electronics booth at the show.

to know what their cycles are, or are they reluctant to have other people see when their highs and lows occur, a privacy thing, perhaps? Anyway, I saw a pretty young lady who couldn't be out of her 20s key in a



Photo 4: Victoria Van Buskirk, 10, with her father Thomas, a programmer at Lockheed in Sunnyvale CA, in the Recreational Computer Centers booth. Tom, with his interests in computer games and home accounting problems, plans to acquire a computer.



Photo 5: The husband and wife team of Ellen and Allen Penn (he's president, she's VP and secretary of Sylvanhills Laboratory Inc, 1 Sylvanway, Box 239, Strafford MO 65757) demonstrate computer controlled plotters to Dick Speer of Northwest Computer Cinema. (For more details on the plotter kit, see BYTE January 1977 page 85.) Dick has a 6502 based computer similar to the KIM which the Penns are using in this setup; he wants to use it for computer controlled graphics.

birth date of October 9 1912. An arbitrary date? The birth date of a parent? Only she knows. The computer took a fair amount of time to figure it all out, but finally traced the patterns on the screen for a 20 day period.

At the Advanced Technology Research Associates booth there were flashing space jewels and Plexiglas ray guns that lit up with a pulsing bright colored light when the trigger was pulled. These fascinated kids and grownups alike, only at 24 bucks and on up the kids weren't buying. The older people were, though. I asked one guy what he planned to use his space ray gun for.

"I'm a magician," he said. "I'll find a use for it on stage."

Another guy wanted to know if the space jewels were waterproof. He thought it might be interesting to drop one of the little flashing pendants into his tropical fish tank, or into a pitcher of his favorite iced beverage. They aren't waterproof; the battery compartment would be flooded.

Like a lot of other people, I had a computer portrait made—you stand in front of a TV camera, your image is held on the TV

Now. THE PERFECT MICROCOMPUTER

VECTOR 1



It's a two touch affair

VECTOR 1 is based on the 8080A Microprocessor and S-100 Bus -

EASY TO ASSEMBLE, EASY TO ACCESSORIZE, EASY TO BUY AND VERY EASY TO OPERATE.

VECTOR GRAPHIC inc. T.M.

717 LAKEFIELD ROAD, SUITE F
WESTLAKE VILLAGE, CA 91361 • (805) 497-0733

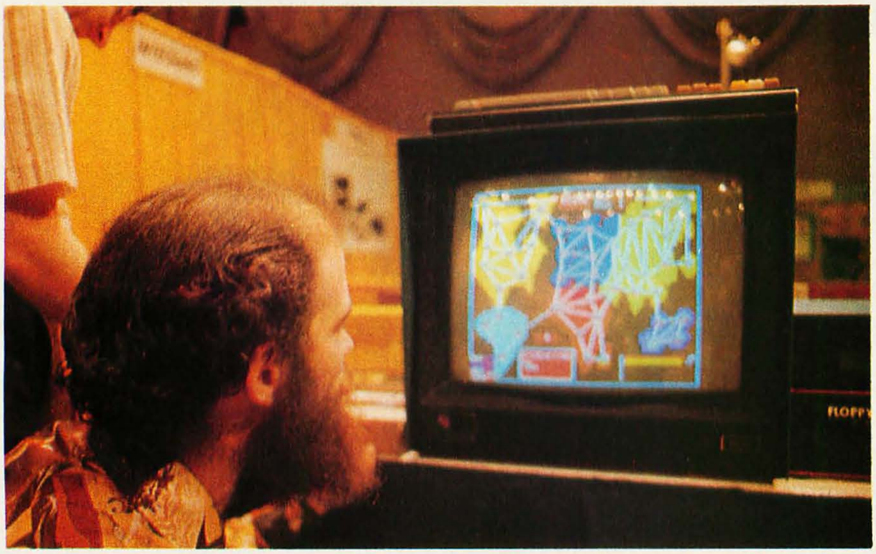


Photo 6: Terry Belcher of Advanced Technology Research studies a map of world conquest for the color video game "Risk" which was being run at the Compucolor booth.

screen until it can be scanned for a computer printout. In about a minute the printer rolls it out and it is torn off and handed to you. The guy in front of me in line made a production out of it. He was bearded, and he wanted a profile shot with a pipe in his mouth and his hand up to it in a studied pose. He explained that he wanted it for his office door.

There were a couple of booths playing computer music and the sound was great, even if the repertoire was a bit limited. About the 15th or so time you've heard a particular tune during the day you can draw the waveforms of each note. But the performance was pretty good. Solid State Music's booth was playing such tunes as "The Flight of the Bumblebee," Bach's "Little Fugue in G Minor," "Blues in F," "It's a Small World," and "Maple Leaf Rag." Try those on your computer.

Footweary after awhile, I went into the cafeteria, got some coffee, and sat down at a table and got into a conversation with two men sitting there. They were Manabu Uyehara and George M Hirota, who have the Radio Shack franchise in Honolulu. They had flown to San Francisco especially for the Faire. Both men have had previous computer experience, and they are anxious to carry computers in their store.

I moved to another table and met Scott M Dickson, a computer programmer in the dental school of the University of Michigan. He wants a computer to experiment with. He thinks dentists could use them for patients' records and for billing.

Coming from the cafeteria I stopped at a booth offering a selection of fine prints and etchings done in sort of a science fiction-fantasy style. One of them, I thought, would look especially fine in my study. Before I could ask how much it was, somebody else did.

"Six-fifty," said the young artist.

Real reasonable, I thought, and moved forward to buy one.

The man who had asked how much, then asked:

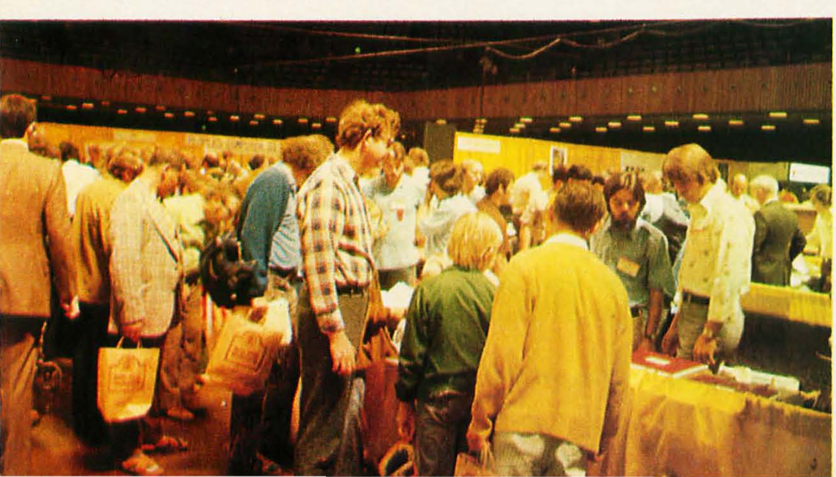
"Six hundred and fifty dollars?"

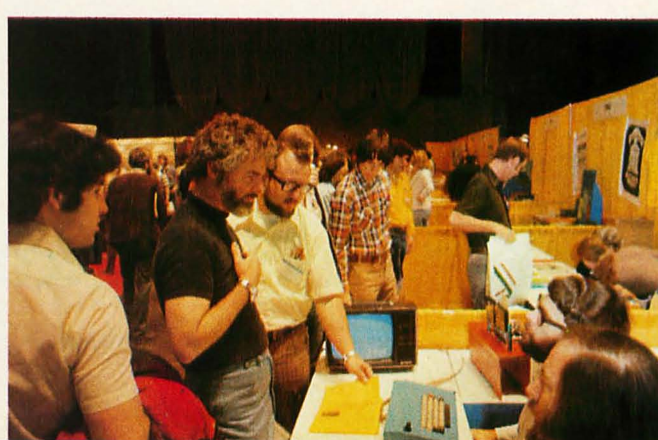
The artist said that was correct.

I walked away.

That's more than most of the computers were selling for.■

Photo 7: Some floor shots.





What's New?

A New 16 Bit Personal Computing System



EBNEK Inc, 254 N Washington St, Wichita KS 67202, has recently announced the new EBNEK 77 system. This system, available for $2770 in kit form or $3800 assembled and tested, features the 16 bit TMS9900 processor manufactured by Texas Instruments Inc. What EBNEK has done is to design a finished package around the TMS9900 with the following characteristics:

- TMS9900 processor board.
- 16 K bytes of programmable memory on two 8 K byte boards.
- 8 K bytes of EROM socket capacity on one board, with 2 K bytes of operating system software supplied (ie: room for 6 K bytes of user defined EROM programs).
- Video display board with television monitor.
- IO interface board with keyboard interface, eight bits of parallel IO, serial data port with data rate generator, and cassette modem.
- Cabinet and power supply motherboard.
- Phi-Deck tape transport with firmware support.
- ASCII keyboard and case.

The design of the system is modular, with an addressing space limitation of 64 K bytes, and a design limit of 128 K bytes using memory bank switching. (Power adequate for a full 128 K bytes is provided as a standard feature according to the company.)

The video display provides 15 lines of either 32 or 64 characters, generated with a 7 by 9 dot matrix. The display also has a point graphics mode allowing a 256 by 240 dot display driven from an 8 K byte segment of main memory. The display and processor are interleaved so that operation of the processor does not break up the picture with visual hash, allowing programmed updates to occur while information is being displayed.

This product looks like a significant one for the personal computing user who wants the ultimate in microprocessor execution speed and performance. The TMS9900 architecture includes 16 general purpose work areas at arbitrary memory locations which use a short form of addressing. It has the unique characteristic of built-in multiply and divide operations which take (worst case without wait states, using 3 MHz clock) 17 and 41 μs to evaluate 32 bit results of 16 bit operands. Individuals with real time computational applications (as in voice recognition, scene analysis, electronic music generation) will find the 9900 architecture quite useful. In addition to the physical configuration, EBNEK's announcement of the EBNEK 77 describes availability of an assembler, editor, utilities and a user's software library.■

Circle 631 on inquiry card.

Polyphonic Music Generator Board

A new interface board designed for music generation called the SRS-320 has been announced by Stillman Research Systems, POB 14036, Phoenix AZ 85063, for $175. The board, which can serve as the basis for a computerized music synthesizing system, is Altair bus compatible and can generate up to four musical notes simultaneously over a four octave range. This is sufficient to generate most elementary chords and inversions. The SRS-320 is priced at $175, and a demonstration cassette of the board is available for $3, as well as a variety of other music generation electronics.■

Circle 632 on inquiry card.

New Vector Catalog Available

Vector Electronic Company Inc's 1977 catalog is now available, listing prices and descriptions of the entire Vector product line, including printed circuit cards, terminals and pins, breadboarding kits, wiring pencils and so on. Of special interest to the computer experimenter is the company's new cordless rechargeable wire wrap tool called Wrap and Strap. The catalog is available free from Vector Electronic Company Inc, 12460 Gladstone Av, Sylmar CA 91342, and no serious experimenter would want to be without it.■

Circle 633 on inquiry card.

Technical Forum

A Proposed Microprocessor Software Standard

Many people are concerned about the need for standardization in the microprocessor industry. One obvious source of standards is people who are seriously designing and supporting microprocessors. The following text was provided by Mostek to document one interchange standard intended for the Mostek version of the Z-80 processor. The text was written by Peter Formaniak and David Leitch and was distributed without copyright. The standard is intended to be an upwards compatible extension of the original Intel 8080 object code format, with provision for relocatable programs and linkage of separate assemblies.

Peter G Formaniak
David Leitch
Mostek
POB 169
Carrollton TX 75006

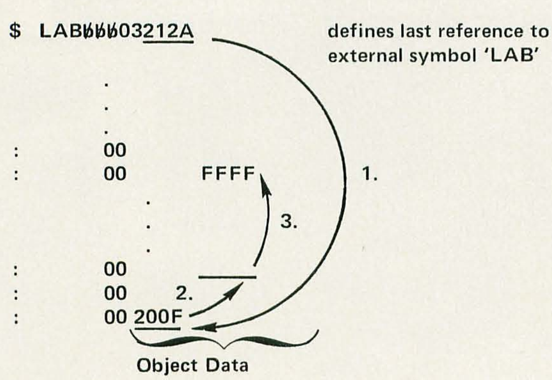
Object Output Definition

Each record of an object module begins with a delimiter (colon or dollar sign) and ends with carriage return and line feed. A colon (:) is used for data records and end of file indicator for Intel compatibility. A dollar sign ($) is used for records containing relocation information and linking information. An Intel loader will ignore such information and allow loading of nonrelocatable, nonlinkable programs. All information is encoded in ASCII format.

Each record is identified by a "type." The type appears in the eighth and ninth bytes of the record and can take five values. In the following text the record type begins each heading.

- Type 00: Data Record

Byte number	Description
1	Colon (:) delimiter.
2, 3	Number of (binary) bytes of data in this record. The maximum is 32 binary bytes (64 ASCII bytes).
4, 5	Most significant byte of start address of data.
6, 7	Least significant byte of start address of data.
8, 9	ASCII zeros. This is the "record type" for data.
10 . . .	data bytes.
Last 2 bytes	Checksum of all bytes except the delimiter, carriage return, and line feed. The checksum is the negative of the binary sum of all bytes in the record.
CRLF	Carriage return, line feed.



- Type 01: End-of-file Record

Byte number	Description
1	Colon (:) delimiter.
2, 3	ASCII zero.
4, 5	Most significant byte of the transfer address of the program. This transfer address appears as an argument of the "END" statement of a program. This Intel standard may prove useful in higher level language programs.
6, 7	Least significant byte of the transfer address.
8, 9	Record type 01.
CRLF	Carriage return, line feed.

- Type 02: Internal Symbol

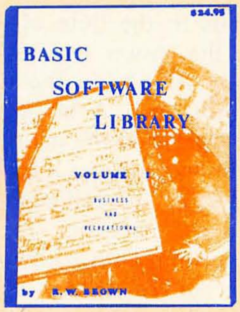
Byte number	Description
1	Dollar sign ($) delimiter.
2, 7	Up to 6 ASCII characters of the internal symbol name. The name is left justified, blank filled.
8, 9	Record type 02.
10, 13	Address of the internal symbol, most significant byte first.
14, 15	Binary checksum.
CRLF	Carriage return, line feed.

Continued on page 62

Fantastic Software

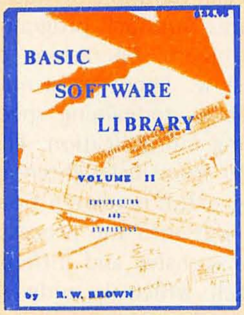
This LIBRARY is a complete do it yourself kit. Knowledge of programming not required. EASY to read and USE

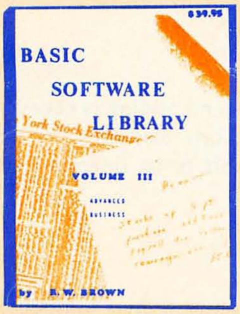
Written in compatible BASIC immediately executable in ANY computer with at least 4K, NO other peripherals needed.



This Library is the most comprehensive work of its kind to date. There are other software books on the market but they are dedicated to computer games. The intention of this work is to allow the average individual the capability to easily perform useful and productive tasks with a computer. All of the programs contained within this Library have been thoroughly tested and executed on several systems. Included with each program is a description of the program, a list of potential users, instructions for execution and possible limitations that may arise when running it on various systems. Listed in the limitation section is the amount of memory that is required to store and execute the program.

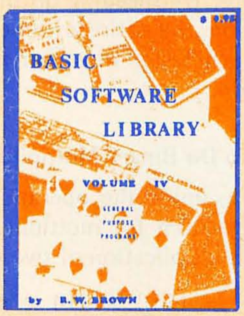
Each program's source code is listed in full detail. These source code listings are not reduced in size but are shown full size for increased readability. Almost every program is self instructing and prompts the user with all required running data. Immediately following the source code listing for most of the programs is a sample executed run of the program.





The entire Library is 1100 pages long, chocked full of program source code, instructions, conversions, memory requirements, examples and much more. ALL are written in compatible BASIC executable in 4K MITS, SPHERE, IMS, SWTPC, PDP, etc. BASIC compilers available for 8080 & 6800 under $10 elsewhere.

This Library is destined to become one of the reference bibles for the small computer field, due to its versatility and uniqueness and the ease of operation of the programs it contains. These volumes are deductible as a business expense when purchased by a company. Send your remittance for prompt delivery, while supplies last. Volume discounts are available to qualified dealers.



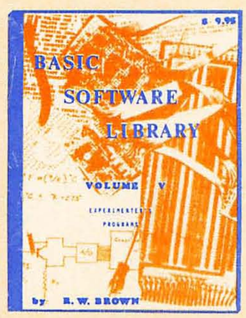
VOLUME I
Bookkeeping
Games
Pictures.................. $24.95

VOLUME II
Math & Engineering
Plotting & Stat
Basic Statement Def........ $24.95

VOLUME III
Advanced Business
Billing
Inventory
Payroll.................. $39.95

VOLUME IV
General Purpose............ $9.95

VOLUME V
Experimenter.............. $9.95



FUTURE ADDITION TO THE "BASIC SOFTWARE LIBRARY"

Volume VI (A Complete Business System - $49.95) General Ledger System - Taxes, Pyrl, W-2's, Inventory, Depr., Financial Statements, etc. AVAILABLE MID SUMMER

Available at most computer stores.
Add $1.50 per volume for postage and handling.

SCIENTIFIC RESEARCH

1712-B FARMINGTON COURT
CROFTON MD 21114

Phone Orders call (800) 638-9194
Information and Maryland Residents Call (301)-721-1148

This Circuit Multiplies

Tom Hall
8500 Cameron Rd
Austin TX 78753

We all know that algorithms for multiplication can be implemented perfectly well in software. But such programmed approaches to arithmetic operations can often prove quite slow. There are a number of possible approaches to gaining speed in operations involving multiplication. These may vary all the way from just tightening up the code to putting a full hardware floating point arithmetic unit into the processor. The latter is often expensive and difficult to build. One approach which is midway between these extremes is to implement the "inner loop" of a software algorithm as some special purpose hardware. An example of this technique is this simple binary multiplier which takes two bytes of information and supplies a 2 byte product value, leaving all the other functions of a full arithmetic package to software. By speeding up the multiplication operation, the entire arithmetic software will run much faster whenever it involves multiplication.

How To Do Binary Multiplication

To examine the operation of a shift and add algorithm for multiplication, let's look at the multiplication of two 4 bit bytes:

```
     1110 =  14 = multiplicand (Y)
    x1010 = x10 = multiplier (X)
    -----
     0000
    1110
   0000
  1110
 --------
 10001100 = 140 = product (P)
```

From inspection it can be seen that the result can be achieved in the same number of steps as there are bits in the byte of the multiplier, and that the answer requires as many bits as the number bits in the two bytes. In other words, two 4 bit bytes generate an 8 bit product, and the multiplication of two 8 bit bytes generates a 16 bit product. Now, by further inspection, it is possible to visualize the necessary implementation of the algorithm with hardware. First, note that the answer requires eight bits. Let's start off then with an 8 bit register (P) set to all zeroes:

```
|0,0,0,0|0,0,0,0| register P
```

Then we'll note that when the first bit to be operated within the multiplier is a zero,

```
|1 0 1 [0]|
```

the result is a zero; so no adding is necessary and we merely shift one place for the next operation. Now the bit to be operated with is a one,

```
|1 0 [1] 0|
```

and one times a number is that number; so we just add the multiplicand into the P register as follows:

```
  |0,0,0,0|0,0,0,0| = register P
  +        1 1 1 0
  -----------------
= |0,0,0,1|1,1,0,0| = new P value
```

Then we shift over one and select the next bit to be operated on:

```
|1 [0] 1 0|
```

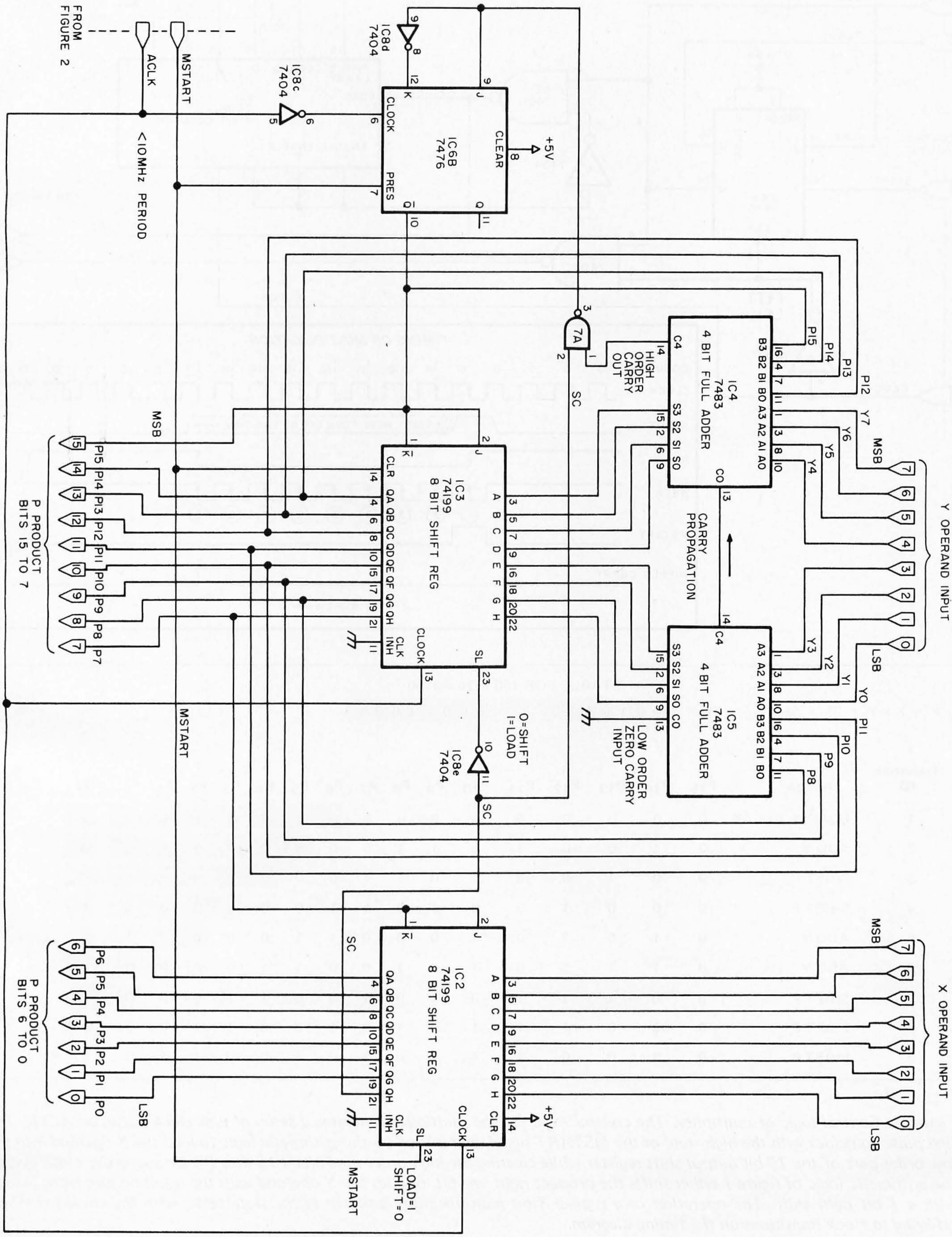
Since it is a zero, only a shift is necessary. Then we operate on the last bit of the multiplier:

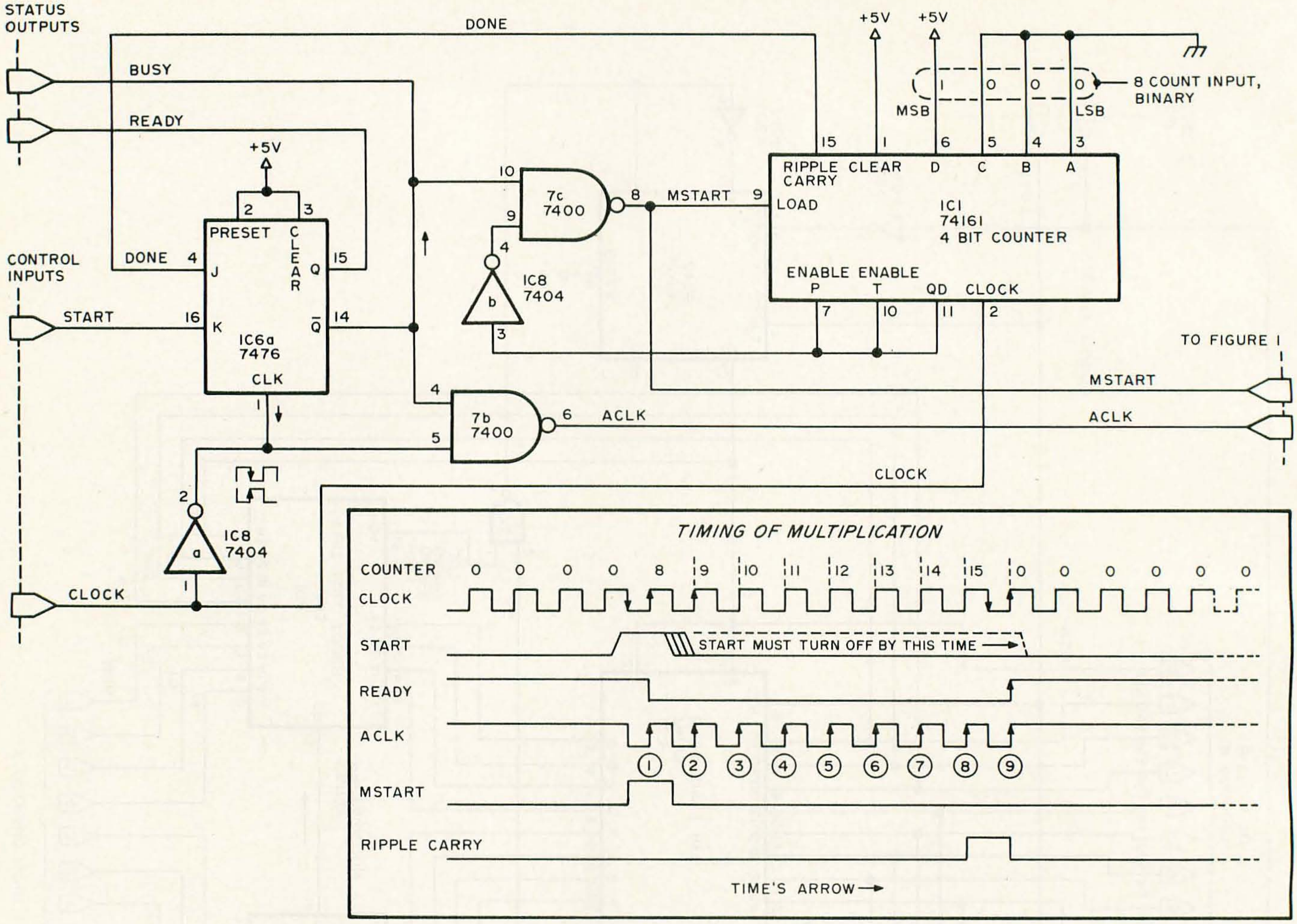
```
|[1] 0 1 0|
```

Table 1: Power wiring for figures 1 and 2.

	Type	+5 V	GND
IC1	74161	16	8
IC2	74199	24	12
IC3	74199	24	12
IC4	7483	5	12
IC5	7483	5	12
IC6	7476	5	13
IC7	7400	14	7
IC8	7404	14	7

Figure 1: Arithmetic logic of multiplier. The X and Y inputs are assumed to be stable throughout the multiplication. The P product output lines are stable after the multiplication is completed. The single bit provided by IC6b extends the product shift register by one bit. Control signals for the operation of the multiplication are ACLK and MSTART generated by the circuit in figure 2.


FROM FIGURE 2
ACLK
MSTART
<10MHz PERIOD
IC8d 7404
IC8c 7404
IC6B 7476
CLEAR
PRES
CLOCK
+5V
7A
SC
IC4 7483 4 BIT FULL ADDER
HIGH ORDER CARRY OUT
CARRY PROPAGATION
IC5 7483 4 BIT FULL ADDER
LOW ORDER ZERO CARRY INPUT
Y OPERAND INPUT
MSB
LSB
IC3 74199 8 BIT SHIFT REG
CLK INH
0=SHIFT 1=LOAD
IC8e 7404
P PRODUCT BITS 15 TO 7
IC2 74199 8 BIT SHIFT REG
X OPERAND INPUT
P PRODUCT BITS 6 TO 0
LOAD=1 SHIFT=0



STATE TABLE FOR 150 X 25 = 3750

P = X x Y = 25 x 150   Y = 1 0 0 1 0 1 1 0   X = 0 0 0 1 1 0 0 1

Transition ID	Action	P_{15}	P_{14}	P_{13}	P_{12}	P_{11}	P_{10}	P_9	P_8	P_7	P_6	P_5	P_4	P_3	P_2	P_1	P_0	SC	
1	LOAD X, CLEAR	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	1	
2	ADD Y	0	1	0	0	1	0	1	1	0	0	0	0	0	1	1	0	0	
3	SHIFT P	0	0	1	0	0	1	0	1	1	0	0	0	0	0	1	1	0	
4	SHIFT P	0	0	0	1	0	0	1	0	1	1	0	0	0	0	0	1	1	
5	ADD Y	0	1	0	1	0	1	0	0	0	1	1	0	0	0	0	0	1	
6	ADD Y	0	1	1	1	0	1	0	1	0	0	1	1	0	0	0	0	0	
7	SHIFT P	0	0	1	1	1	0	1	0	1	0	0	1	1	0	0	0	0	
8	SHIFT P	0	0	0	1	1	1	0	1	0	1	0	0	1	1	0	0	0	
9	SHIFT P	0	0	0	0	1	1	1	0	1	0	1	0	0	1	1	0	0	P = 3750

Figure 2: Control logic of multiplier. The control logic for the multiplier generates a series of nine clock pulses on ACLK. The first pulse coincides with the high state of the MSTART signal, and is used by the arithmetic logic to load the X operand into the low order part of the 17 bit output shift register while clearing the high order nine bits, P15 thru P7. On successive clock cycles, the arithmetic logic of figure 1 either shifts the product right one bit, or adds the Y operand with the resulting sum being loaded with a 1 bit right shift. The operation of a typical 8 bit multiplication is shown in the state table, with the circled numbers referring to clock transitions on the timing diagram.

This is a one; so we again add the multiplicand into the P register into the position indicated by the shift:

```
  |0,0,0,1|1,1,0,0|   register P
+    1 1 1 0
  ---------------
  |1,0,0,0|1,1,0,0|   = final product
                        in P register
```

Thus, the product is generated.

How It Works

The multiplication logic described for the 4 bit case is generalized to 8 bit operands with a 16 bit product in the circuit of figures 1 and 2. The multiplier logic of this design accepts two 8 bit bytes, X and Y, from which it generates the 16 bit product P. After the start signal, a mere ten clock periods suffice to generate the product value. Since TTL devices are utilized, it is possible to generate the result in less than one microsecond by using a fast enough clock. Practically speaking though, with the 1 MHz speed of the typical microprocessor clocks, a result in about ten microseconds will be quite acceptable, especially when comparison is made to the equivalent pure software approach.

The multiplier unit provides optional ready and busy lines as status bits for the processor, and requires a free running clock plus a start signal from the processor. The ready and busy lines can be ignored entirely in most computers, by simply making sure that one or two instructions are executed, taking ten clock periods total between start of a multiply and reference to the product value.

The Multiplication Cycle

Let us assume that the unit is idle, waiting for an input. This idling situation is shown in the timing diagram of figure 2 at the left-hand side. At idle, the BUSY output will be low and the READY output will be high. To start the multiplication, a high level is placed on the START input *after* the X and Y bytes are set up and presented to the unit. At the low to high edge transition of the next clock pulse after the START line is raised, the READY and BUSY lines will change state as flip flop IC6a is clocked. Within eight clock times of these transitions, the START level should be brought low again to keep the unit from restarting the multiplication upon completion of operation. These transitions raise the MSTART line to a high level, which causes the control counter IC1 to be loaded with the value 8 (binary 1000 on the D,C,B,A inputs). At this time, the X input byte is loaded into shift register IC2, and the high order product bits in shift register IC3 and flip flop IC6b are all cleared. All this action happens on the ACLK transition, labelled 1 in the timing diagram of figure 2.

The shift and add multiplication algorithm occurs during the next 8 clock transitions, labelled 2 thru 9 in the timing diagram. At the low to high transition of these clocks the binary counter IC1 is incremented, and a shift right occurs in the shift register IC2 which holds P0 to P6 of the final product and starts out with the value of X. If the shift carry output (SC) of IC2 is low, the transition will simply shift the high order portions of the product (IC6b and IC3) right by one bit, since binary 0 has no effect on the sum. But if an X bit was 1, SC will be high, which sets up control lines so that Y is added to the previous value of the product P and loaded into P with a right shift 1 bit position. (This right shift is in the wiring of the adder as shown in figure 1, where, for example, the old bit P15 plus the high order bit Y7 of Y are added together to form bit P14 of the new P value, and the new P15 is taken from the carry out of the addition process.) The operation of the algorithm is shown in the state table of figure 2 for the example 25 x 150 = 3750.

The multiplication operation is terminated when the binary counter reaches a count of 15, thus generating a ripple carry output which resets the flip flop IC6a to a READY state at the final high to low transition of the ACLOCK clock during a multiplication. If the external circuitry tests READY, this indicates end of the multiplication and a valid product on the P output lines. Note that the binary counter performs one more increment to a zero count, then inhibits itself until the next START signal.■

A Memory Address Space Interface

One good way to interface the multiplier is through the wiring of two addresses with appropriate output latches and decoding logic. Each address has a possible input and output significance, corresponding to whether the processor is reading or writing data to the "memory" location which is the multiplier:

Address	Write Action	Read Action
n	Load X operand latch	Read P_{15} to P_8
n+1	Load Y operand latch and start multiply.	Read P_7 to P_0

This method of interfacing avoids using IO instructions, and can take advantage of the double byte load and store instructions of the 8080 or 6800. (In certain cases, such as array indexed address calculation, the 8080's double byte addition operation can prove useful.)

Announcing a Computer that thinks in BASIC for only $298.



Ohio Scientific's new Model 500 computer has full mini-computer BASIC in ROM and a minimum of 4K RAM for user programs. BASIC is always there when the computer is turned on.

Make your terminal intelligent!

If your company or school has remote computer terminals, consider the Model 500-1. It fits in the communications line between the terminal and modem or system. When the 500-1 is off, the terminal talks to the modem. When it is on, the terminal talks to the Model 500. Use the Model 500 for short and medium sized programs in BASIC. Use its *immediate* mode as an ultra-powerful scientific calculator!

Computers for Students

Couple a Model 500 to a low cost purchased or leased terminal such as the popular ASR-33 teletype for student instruction. The Model 500's BASIC is a perfect instructional language for students. Since the computer's machine code is also accessible, the Model 500 can challenge the advanced student with other optional software such as our interactive Assembler/Editor.

Personal Computing

The Model 500 has eliminated all of the grief of the first generation personal computers. This computer comes fully assembled, tested, and guaranteed.
Most importantly, it is *easy to use* since the BASIC is always there.

Flexibility and Expandability

The Model 500 uses Ohio Scientific's standard 48 line bus so that it is fully compatible with our Model 400 kits and Challenger™ product lines. Any of thirteen accessory boards including RAM, PROM, parallel, serial, A/D, D/A, cassette, and video graphics can be added as well as peripherals including floppy disks, line printers, color graphics, and more.

The Model 500 is offered here in three forms to meet every application and pocket book.

Specifications

Model 500 Board $298.00

8″ x 10″ fully assembled board including 6502 microprocessor running at 1MHz, 512 bytes of PROM, 8192 bytes of ROM containing 8K BASIC and 4096 bytes of RAM for user programs. The board contains an ACIA based serial interface which is jumper selectable for RS-232C or 20ma loop at 110, 300, 1200, 2400, or 4800 baud.

The 8K BASIC features: full floating point math including transcendental functions, N dimension arrays, multiple letter variables, full string functions, logical operators, PEEK, POKE, USR, and lots more!

The Model 500 board requires +5 volts at 2 amps, −9 volts at 500ma, an external reset switch, and an ASCII serial terminal for operation.

Model 500-1 $429.00

Fully enclosed 500 board with power supply, reset switch, and two 25 pin EIA standard terminal connectors for loop through operation. The 500-1 is 12″ x 15″ x 4″.

Model 500-8 $629.00

This unit is a 500 board in an eight slot Challenger™ case allowing seven slots for expansion. The unit has a heavy-duty UL recognized power supply and is 15″ x 17″ x 10″.

ORDER FORM

Order directly from:
Ohio Scientific; Hiram, Ohio 44234

This introductory offer expires August 31, 1977. Please allow up to 60 days for delivery.

PRINT OR TYPE CLEARLY.

Name ______

Shipping Address ______

City ______ State ______ Zip ______

Payment by: BAC (VISA) ___ MC ___ Money Order ___

Credit Card Account # ______ Expires ______

Interbank # (Master Charge) ______

___ Model 500 Boards @ $298.00 ______

___ Model 500-1 Computer @ $429.00 ______

___ Model 500-8 Computer @ $629.00 ______

___ Additional 4K RAM Memories for 500-8 @ $149.00 ______

4% Sales Tax (Applies to Ohio Residents only) ______

TOTAL CHARGED OR ENCLOSED ______

All orders shipped insured UPS unless otherwise requested.

Circle 12 on inquiry card.

OHIO SCIENTIFIC

Hiram, Ohio 44234

Ask BYTE

S-100 BUS?

Would you please show the pin description of the S-100 bus, or at least tell me where I can find it?

David W Anderson
15500 S 1300 W
Bluffdale UT 84065

The S-100 bus, so called, is the name used by "everybody else" who manufactures a MITS Altair compatible peripheral, processor or board. MITS Inc defined the bus by designing the Altair 8800, and during 1976 various manufacturers who make Altair compatible peripherals and processors with Altair-like backplanes began calling it the "S-100" bus in advertisements. So for detailed information, purchase the documentation of an Altair 8800, or one of the other central processors which copy this arrangement. Sooner or later, we expect to print an article detailing this information. ■

MULTIPLICATION

. . . In the April 1977 BYTE you published some information on the 8008. From looking at the instruction set published I don't see a multiplication instruction. Do any microprocessors have a hardware multiply instruction, or is it always in software?

Joseph H Stockton Jr
Rt 6, Caroline Dr
Stockbridge GA 30281

The 8008 was the first microprocessor to be both widely marketed and of sufficient capability to make a decent general purpose computer. Since it came out, in the early years of this decade, designs have evolved considerably, but few designs to date (1977, first quarter) include hardware multiply and divide functions. The Digital Equipment Corporation's LSI-11 has a PDP-11 instruction set which does not include multiply or divide operations in its basic set, but which has an optional extended arithmetic option which provides 16 bit fixed point arithmetic operations and 32 bit floating point arithmetic operations. The Texas Instruments TMS-9900 processor has multiply and divide operations for 16 bit integer arithmetic built into its design. But the popularly used 8080, Z-80, 6800 and 6502 processors do not incorporate this feature and the user must do multiplication and division in software. ■

Condensed Reference Chart

$^{a}NN^{b}$
Mne. Mode
Cond. Flags
OPERATION

Figure 1: This is the basic four line format that is used to describe each particular opcode instruction. See text for a detailed description of the notation used.

Robert J Borrmann
Electrical Engineering Dept
Manhattan College
Riverdale NY 10471

Here is a quick reference chart that describes the opcodes for the Motorola 6800 microprocessor. This chart gives all of the hexadecimal codes, including the ones that are unimplemented. The codes are given in a four line format as shown in figure 1.

The first line has three values: a, b and NN. NN is the hexadecimal op code; a is the length, one, two or three bytes, of the instruction; b is the length of execution in machine cycles for each instruction.

The third line contains the condition flags that are affected by the execution of the instruction. The notation used for the condition flags is summarized in table 2. If a flag is set to a specific value by the instruction, the flag and value are shown on line 4. If the command is a branch instruction then this line contains the conditions under which the branch will occur in parentheses. The letter "u" represents an operand interpreted as an unsigned 8 bit binary number, "s" represents an operand interpreted as a signed two's complement number. The "u" and "s" conditions are correct only if the last flag affecting instruction executed before the branch is SBA, CBA, SUB or CMP.

The fourth line contains a summary of the operations of the instruction in the AHPL language. AHPL stands for A Hardware Programming Language and is discussed by Hill and Peterson in *Digital Systems: Hardware Organization and Design* which is published by John Wiley and Sons Inc. The notation is taken from page 112 of that book. The upward and downward pointing arrows stand for various types of shifts. The upward pointing arrow is a circular shift left and the downward pointing arrow is a circular shift right. An arrow with a circle at its beginning denotes a shift with a zero moving into the vacated end. The arrow with the ditto mark (") is my own invention to denote a shift with the vacated end bit maintained. The ⊕ symbol is used to denote the exclusive OR function. The left pointing arrow is the APL standard assignment symbol. If a condition flag is set to a particular value by the instruction, the value is indicated. Any condition flags that do not have a particular notation follow the ordinary rules. Primes on the flag condition, such as v́ in the LSR, ROR and ASR instruction, denote a special rule for setting or clearing that flag.

A nonstandard notation concerns the pushing or pulling of the stack. Pulling from the stack is accompanied by an increase in the value of the stack pointer. The value is increased by 1 in the PULA and PULB instructions; by 2 in the RTS instruction; and by 7 in the RTI instruction. Similarly, pushing into the stack is accompanied by a decrease in the value of the stack pointer by the appropriate amount.■

A = Accumulator A.
B = Accumulator B.
\# = Immediate.
d = Direct.
x = Indexed.
e = Extended.
r = Relative.
= Inherent (no symbol).

Table 1: Summary of the addressing modes that are used in the chart. These modes are found in blue on the chart.

CCR = xxhinzvc.
h = Half carry.
i = Interrupt mask.
n = Negative.
z = Zero.
v = Overflow.
c = Carry.
x = Don't Care, always 1

Table 2: Summary of the condition flag codes that are used in the chart.

Editor's Note . . .
The Holes in Opcode Space

Looking at this chart of the 6800 (or a similar chart for any processor) one sees numerous "holes" in operation code space. These holes are the yellow boxes. An interesting challenge for the owner of a microprocessor (or any computer) is to try and figure out what happens when the forbidden opcodes are executed. Will these exotica turn out to have any uses? Or will they simply be another set of NOPs? For example, look at the strange grouping of the operation codes 4X, 5X, 6X, 7X where X is any hexadecimal digit. 40, 50, 60, 70 are all negates, 43, 53, 63, 73 are all complements, and various arithmetic operations occur for other values. But the operation codes 41, 51, 61 and 71 are nominally undefined as are 4B, 5B, 6B, 7B and several others. Could it be that those are operation codes that weren't documented in the literature because the masks of the LSI chip had errors? Could it be that they are supposedly unimplemented but are really just not documented so that a "next generation" processor can be achieved by simply releasing the paperwork and selling "new" chips? It's an area of personal research which could prove quite useful: Whenever you find a new processor, examine it for missing opcodes and see what wasn't documented — then try to figure it out by writing programs which execute "unimplemented" instructions under carefully controlled conditions.

for the 6800

0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	^{1}10^{2} SBA nzvc A←A−B	^{2}20^{4} BRA r PC←PC+r+2	^{1}30^{4} TSX IX←SP+1	^{1}40^{2} NEG A nzvc OP←¯OP	^{1}50^{2} B	^{2}60^{7} X	^{3}70^{6} e	^{2}80^{2} SUB A# nzvc ACC←ACC−OP	^{2}90^{3} Ad	^{2}A0^{5} AX	^{3}B0^{4} Ae	^{2}C0^{2} B#	^{2}D0^{3} Bd	^{2}E0^{5} BX	^{3}F0^{4} Be
^{1}01^{2} NOP	^{1}11^{2} CBA nzvc nzvc←f(A−B)	21	^{1}31^{4} INS SP←SP+1	41	51	61	71	^{2}81^{2} CMP A# nzvc nzvc←f(ACC−OP)	^{2}91^{3} Ad	^{2}A1^{5} AX	^{3}B1^{4} Ae	^{2}C1^{2} B#	^{2}D1^{3} Bd	^{2}E1^{5} BX	^{3}F1^{4} Be
02	12	^{2}22^{4} BHI r (U1>U2)	^{1}32^{4} PULA A←TOS	42	52	62	72	^{2}82^{2} SBC A# nzvc ACC←ACC−OP−c	^{2}92^{3} Ad	^{2}A2^{5} AX	^{3}B2^{4} Ae	^{2}C2^{2} B#	^{2}D2^{3} Bd	^{2}E2^{5} BX	^{3}F2^{4} Be
03	13	^{2}23^{4} BLS r (U1⩽U2)	^{1}33^{4} PULB B←TOS	^{1}43^{2} COM A nzvc OP←~OP v←0 c←1	^{1}53^{2} B	^{2}63^{7} X	^{3}73^{6} e	83	93	A3	B3	C3	D3	E3	F3
04	14	^{2}24^{4} BCC r (c=0)	^{1}34^{4} DES SP←SP−1	^{1}44^{2} LSR A nzv́ć (OP,c)←↓(OP,c) n←0	^{1}54^{2} B	^{2}64^{7} X	^{3}74^{6} e	^{2}84^{2} AND A# nzv ACC←ACC∧OP, v←0	^{2}94^{3} Ad	^{2}A4^{5} AX	^{3}B4^{4} Ae	^{2}C4^{2} B#	^{2}D4^{3} Bd	^{2}E4^{5} BX	^{3}F4^{4} Be
05	15	^{2}25^{4} BCS r (c=1)	^{1}35^{4} TXS SP←IX−1	45	55	65	75	^{2}85^{2} BIT A# nzv nz←f(ACC∧OP), v←0	^{2}95^{3} Ad	^{2}A5^{5} AX	^{3}B5^{4} Ae	^{2}C5^{2} B#	^{2}D5^{3} Bd	^{2}E5^{5} BX	^{3}F5^{4} Be
^{1}06^{2} TAP ihnzvc CCR←A	^{1}16^{2} TAB nzv B←A, v←0	^{2}26^{4} BNE r (z=0)	^{1}36^{4} PSHA TOS←A	^{1}46^{2} ROR A nzv́ć (OP,c)←↓(OP,c)	^{1}56^{2} B	^{2}66^{7} X	^{3}76^{6} e	^{2}86^{2} LDA A# nzv ACC←OP v←0	^{2}96^{3} Ad	^{2}A6^{5} AX	^{3}B6^{4} Ae	^{2}C6^{2} B#	^{2}D6^{3} Bd	^{2}E6^{5} BX	^{3}F6^{4} Be
^{1}07^{2} TPA A←CCR	^{1}17^{2} TBA nzv A←B, v←0	^{2}27^{4} BEQ r (z=1)	^{1}37^{4} PSHB TOS←B	^{1}47^{2} ASR A nzv́ć (OP,c)←↓(OP,c)	^{1}57^{2} B	^{2}67^{7} X	^{3}77^{6} e	87	^{2}97^{4} STA Ad nzv OP←ACC, v←0	^{2}A7^{6} AX	^{3}B7^{5} Ae	C7	^{2}D7^{4} STA Bd nzv OP←ACC, v←0	^{2}E7^{6} BX	^{3}F7^{5} Be
^{1}08^{4} INX z IX←IX+1	18	^{2}28^{4} BVC r (v=0)	38	^{1}48^{2} ASL A nzv́ć (c,OP)←↑(c,OP)	^{1}58^{2} B	^{2}68^{7} X	^{3}78^{6} e	^{2}88^{2} EOR A# nzv ACC←ACC⊕OP, v←0	^{2}98^{3} Ad	^{2}A8^{5} AX	^{3}B8^{4} Ae	^{2}C8^{2} B#	^{2}D8^{3} Bd	^{2}E8^{5} BX	^{3}F8^{4} Be
^{1}09^{4} DEX z IX←IX−1	^{1}19^{2} DAA nzv́ć A,CCR←( )	^{2}29^{4} BVS r (v=1)	^{1}39^{5} RTS PC←DTOS	^{1}49^{2} ROL A nzv́ć (c,OP)←↑(c,OP)	^{1}59^{2} B	^{2}69^{7} X	^{3}79^{6} e	^{2}89^{2} ADC A# hnzvc ACC←ACC+OP+c	^{2}99^{3} Ad	^{2}A9^{5} AX	^{3}B9^{4} Ae	^{2}C9^{2} B#	^{2}D9^{3} Bd	^{2}E9^{5} BX	^{3}F9^{4} Be
^{1}0A^{2} CLV v v←0	1A	^{2}2A^{4} BPL r (n=0)	3A	^{1}4A^{2} DEC A nzv OP←OP−1	^{1}5A^{2} B	^{2}6A^{7} X	^{3}7A^{6} e	^{2}8A^{2} ORA A# nzv ACC←ACC∨OP, v←0	^{2}9A^{3} Ad	^{2}AA^{5} AX	^{3}BA^{4} Ae	^{2}CA^{2} B#	^{2}DA^{3} Bd	^{2}EA^{5} BX	^{3}FA^{4} Be
^{1}0B^{2} SEV v v←1	^{1}1B^{2} ABA hnzvc A←A+B	^{2}2B^{4} BMI r (n=1)	^{1}3B^{10} RTI ihnzvc Rg←7S+K	4B	5B	6B	7B	^{2}8B^{2} ADD A# hnzvc ACC←ACC+OP	^{2}9B^{3} Ad	^{2}AB^{5} AX	^{3}BB^{4} Ae	^{2}CB^{2} B#	^{2}DB^{3} Bd	^{2}EB^{5} BX	^{3}FB^{4} Be
^{1}0C^{2} CLC c c←0	1C	^{2}2C^{4} BGE r (S1⩾S2)	3C	^{1}4C^{2} INC A nzv OP←OP+1	^{1}5C^{2} B	^{2}6C^{7} X	^{3}7C^{6} e	^{3}8C^{3} CPX # ńzv́ nzv←f(IX−DOP)	^{2}9C^{4} d	^{2}AC^{6} X	^{3}BC^{5} e	CC	DC	EC	FC
^{1}0D^{2} SEC c c←1	1D	^{2}2D^{4} BLT r (S1<S2)	3D	^{1}4D^{2} TST A nzvc nz←f(OP) v←0 c←o	^{1}5D^{2} B	^{2}6D^{7} X	^{3}7D^{6} e	^{2}8D^{8} BSR r	9D	^{2}AD^{8} JSR X	^{3}BD^{9} e	CD	DD	ED	FD
^{1}0E^{2} CLI i i←0	1E	^{2}2E^{4} BGT r (S1>S2)	^{1}3E^{9} WAI 7Stk←Rg	4E	5E	^{2}6E^{4} JMP X PC←OP	^{3}7E^{3} e	^{3}8E^{3} LDS # nzv SP←DOP, v←0	^{2}9E^{4} d	^{2}AE^{6} X	^{3}BE^{5} e	^{3}CE^{3} LDX # nzv IX←DOP, v←0	^{2}DE^{4} d	^{2}EE^{6} X	^{3}FE^{5} e
^{1}0F^{2} SEI i i←1	1F	^{2}2F^{4} BLE r (S1⩽S2)	^{1}3F^{12} SWI i←1 7Stk←Rg	^{1}4F^{2} CLR A nzvc OP←0,n←0,z←1,v←0,c←0	^{1}5F^{2} B	^{2}6F^{7} X	^{3}7F^{6} e	8F	^{2}9F^{5} STS d nzv DOP←SP, v←0	^{2}AF^{7} X	^{3}BF^{6} e	CF	^{2}DF^{5} STX d znv DOP←IX,v←0	^{2}EF^{7} X	^{3}FF^{6} e

A Train Control Display Using the LSI-11 Microcomputer

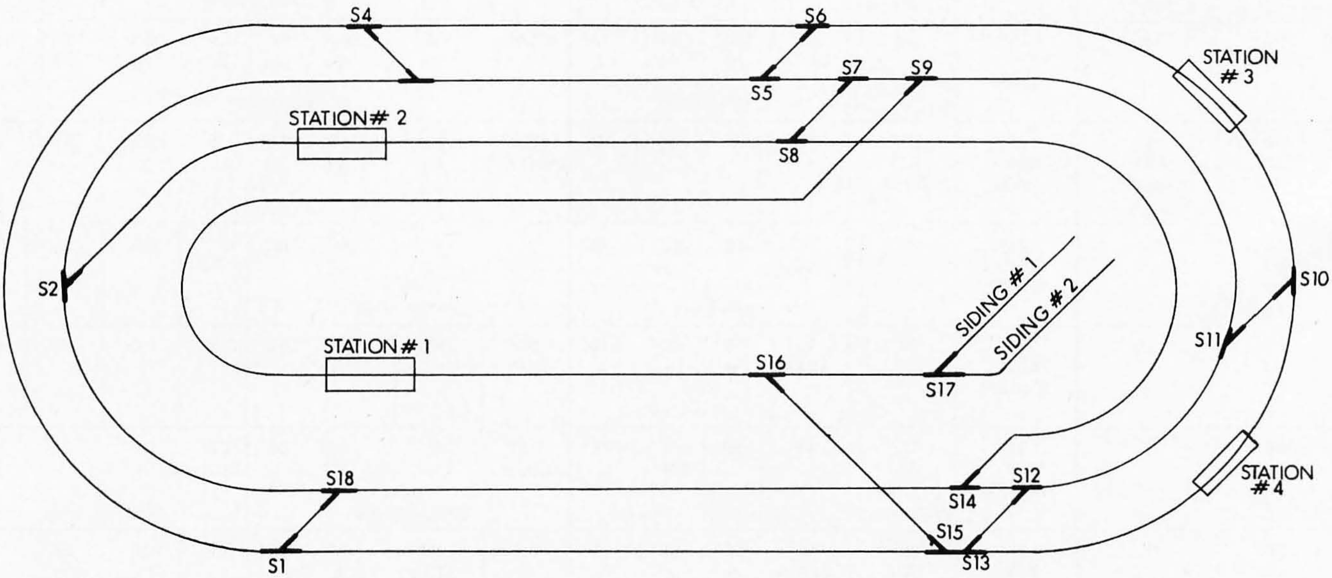
Jack Hart
Digital Components Group Engineering
Ed Badger
System Diagnostic Engineering
Digital Equipment Corporation
Marlborough MA 01752

Figure 1: Model railroad track layout used in authors' system. During a typical demonstration, both trains leave their sidings under control of a Digital Equipment Corporation LSI-11 microcomputer and are routed along the tracks according to a control program in the computer. Feedback sensors are used to verify train locations at strategic times.

In order to demonstrate the application of a microcomputer to real world situations, a model train system was developed which is controlled by a Digital Equipment Corporation LSI-11 microcomputer. Two model trains were used for the demonstration — one adhered to a fixed schedule while the other was directed along a path to make it avoid the first train.

HO model railroad components were used. (HO is currently the most popular size for model trains and accessories among enthusiasts in the United States; the scale is 1/64th of actual size, but this can vary.) The layout includes some 75 feet of track and 18 switches plus detectors to determine train locations and a programmable power supply to control train speed and direction. Control interfaces are used between the computer and the track for power. In addition, switches are used to control train position and both station indicators and billboards check train positions and provide atmosphere. A 60 Hz clock is used for a time base, and peripheral devices are used for data entry, status reporting and transferring commands to the train system.

As in many other such systems, the hardware and software elements of this system can be analyzed separately, but it is important to keep the interrelations between the two in mind. Thus, when a detector informs the computer that a train is in a certain location, the control program must update the



TRACK LAYOUT INDICATING 18 SWITCHES, 4 STATIONS AND 2 SIDINGS

system's files so that the new location can be related to the location of the other train.

Take the A Train

The operation of this electric train system is straightforward (see figures 1 and 2). Upon startup, the trains are moved forward until they trip the location sensors in their respective sidings. These location sensors are actually reed switches which will close in the presence of a magnetic field. Each train is provided with a magnet for this purpose (the next section describes these switches in greater detail). After both siding sensors have been tripped, the trains (called A and B for convenience) are individually brought clear of the siding area.

The lead train A is switched to a station and train B is diverted to the outer track. The software used to control the trains and keep them separated is then employed throughout the remainder of the LSI-11 demonstration.

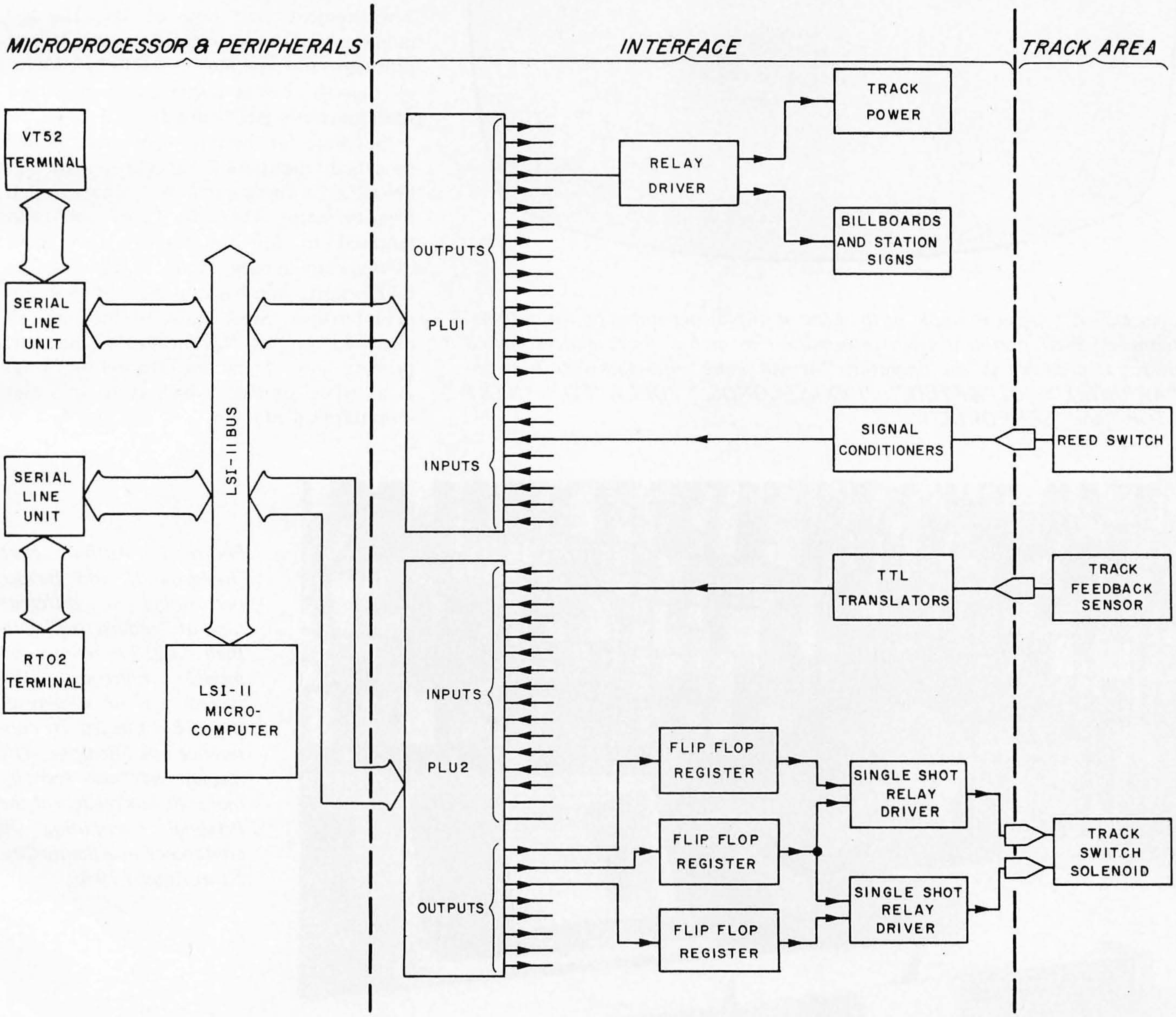


Figure 2: Block diagram of the LSI-11 control system. The data bus is connected to two terminals used to input and display information at the operator's control station. It is also connected to a special interface which is based on two hardware latches called parallel line units, or PLUs. These PLUs store command bytes coming from the microcomputer which in turn drive flip flops and relay drivers used to power the track and operate track switches. Commands from the computer are in the form of 16 bit words. Each section of the track has been assigned a specific bit, and only those bits of the word necessary to change the track sections desired will change. Information from reed switches is also latched by the PLUs and sent back to the computer to be used in closing the feedback control loop.

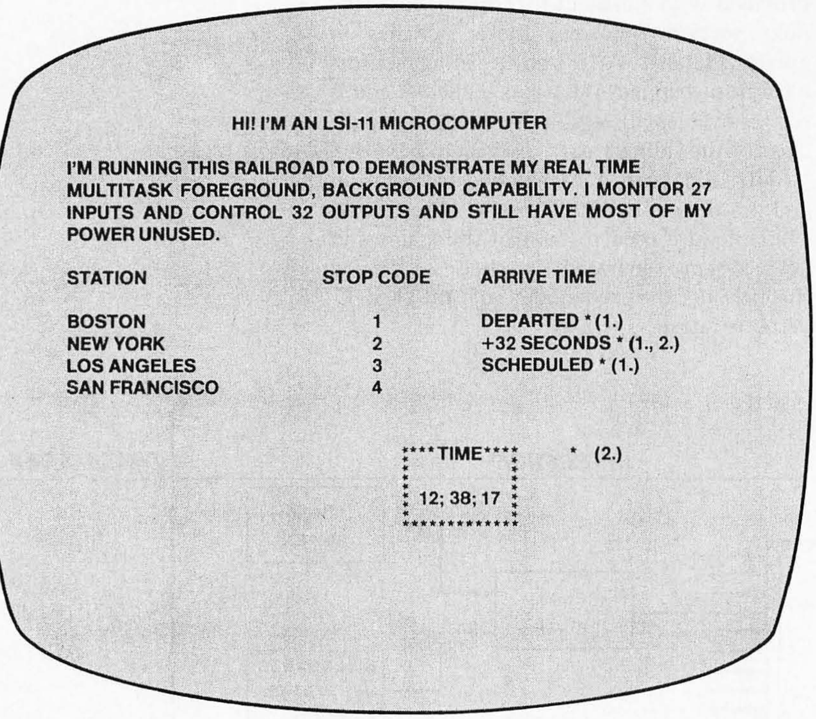


Figure 3: A typeset example of the type of display generated by the authors' terminal. Such pertinent train information as arrival times and estimated delays is available to the operator. "Arrival Time" messages vary between "ARRIVED," "DEPARTED," "+XX SECONDS," "DELAYED," "NEXT STOP," and "SCHEDULED."

Operator instructions such as specifying where the trains are to stop and requests from the computer for instructions are transmitted by an RT02 remote data terminal. The terminal has a 30 character keyboard input and can display as many as 32 alphanumeric characters. These characteristics provide all the capabilities needed for full interaction between the operator and the system. A full video terminal, the VT52, is used to display train information as well as advertisements and time of day. The terminal is serially connected to the central processor and operates at 9,600 bps. Multiple messages can be displayed on the screen simultaneously (see figure 3).

Software for the train demonstration can be called "multitask." Tasks are assigned to be either a foreground or a background priority status. There are 12 possible tasks required to run the train system, eight of which are in background. Foreground jobs take priority over background jobs by means of interrupts. Since most of the jobs are executed on an "as needed" basis, the primary job of the background is to act as a "wheel spinner" which cycles in a wait loop until called upon.



Photo 1: Authors Hart (foreground) and Badger performing a different kind of "engineering" with their LSI-11 controlled model railroad. The system was on display at the IEEE Electro 76 conference in Boston. The display was also seen by many of our readers at the Personal Computing 76 conference in Atlantic City NJ in August 1976.

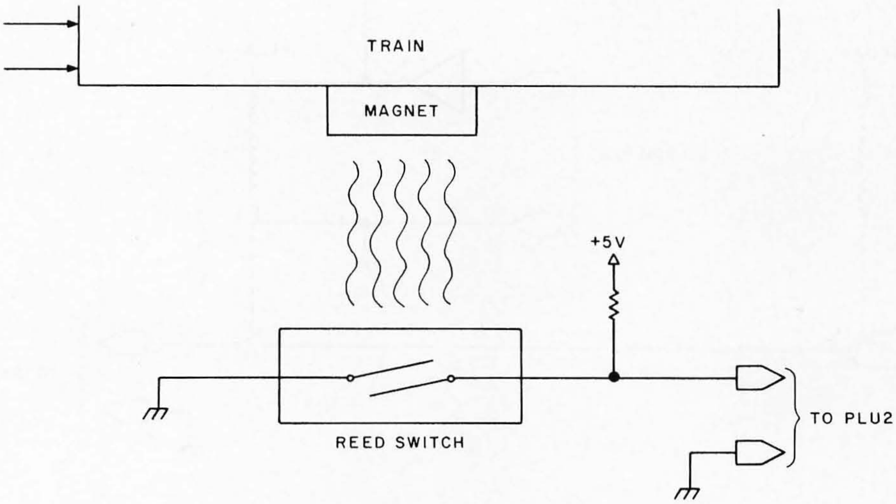


Figure 4: Reed switch circuitry used to send feedback information to the microcomputer. When a train passes by, the magnetic field causes the switch to close, pulling the signal line down to ground potential; this sends a signal back to the interface (and finally to the computer) that a train is passing.

Software execution in this system is controlled by a real time clock program module which creates a reference base to coordinate all program operations. This particular clock makes use of a standard 60 Hz power line frequency which is passed through a low pass noise filter and then transformed into a train of square waves at TTL voltage level. These square waves are next fed to the BUS event line of the LSI-11 where they create processor interrupts at each positive going transition of the waveform. Every 16.7 ms, then, control is transferred to the clock job, which regulates the "time of day" and "train arrival" displays on the video terminal.

Keeping Track of Your Trains

Neither train can be successfully controlled unless its location is known with a great deal of certainty at any given time. This is done by processing information from the various reed switches and extrapolating times of arrival at key points (eg: when train A is scheduled to arrive at a station).

As mentioned earlier, each train has a magnet mounted on it to activate the reed switch sensors (see figure 4). Each sensor is polled periodically to determine if a train is present. From this information it is possible to compare train position against the schedule and the time base to quickly tell if a train is on time or late; this information can be displayed on the video screen. The reed switches are connected to the input side of parallel line unit number one (PLU 1, which serves as a data storage buffer to and from the computer) via a TTL translator which converts the switches' status to the appropriate TTL voltage levels needed by the PLU. PLU 1's output is used to energize the billboard signs and sections of track under control of the LSI-11.

Powering Up

When two trains are to be controlled simultaneously, as in this case, the simplest technique is to apply power to isolated sections of track at the right times. The computer can differentiate between trains A and B by checking position and schedule and

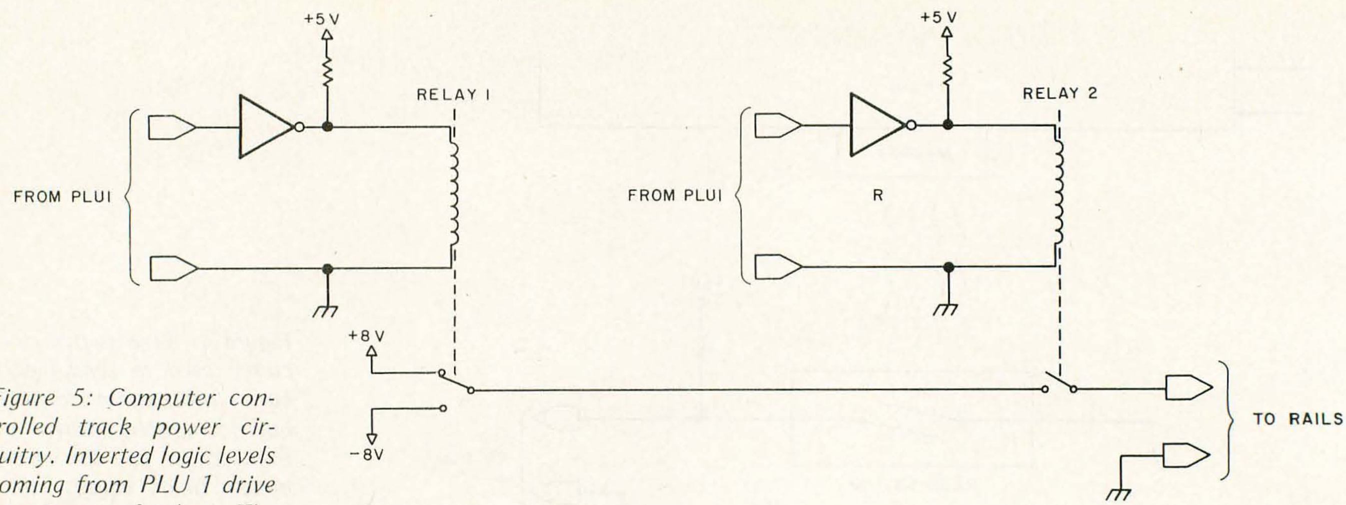


Figure 5: Computer controlled track power circuitry. Inverted logic levels coming from PLU 1 drive two types of relays. The first type (represented by relay 1 in this illustration) chooses the polarity of the voltage going to the track, while the second type (relay 2 here) applies power to the track. Only certain strategic areas of the track require polarity switching.

then creating the proper track energization sequence to control the trains. There is one problem in using PLU 1's output to energize the tracks: its TTL output power is too low to drive the HO trains being used. This problem can be solved by the use of relays (see figure 5).

Relaying Information

Two types of relays are employed. For those sections of track which are simply to be turned on or off, a single pole relay is sufficient. When power is required, the appropriate output word is transferred to PLU 1, which causes the desired relay to be energized. This in turn applies power to the track. When power is no longer required, PLU 1 is updated with a new word and the relay is de-energized. Some tracks, however, have reversing capabilities which require not only power but also one of two polarities for track voltage. Figure 5 shows how two relays are used to do this: relay 1 chooses the polarity of the track based on a command from PLU 1, while relay 2 applies power to the track, again on a command from PLU 1. Each time a command is given, a single 16 bit word is written into the output data buffer

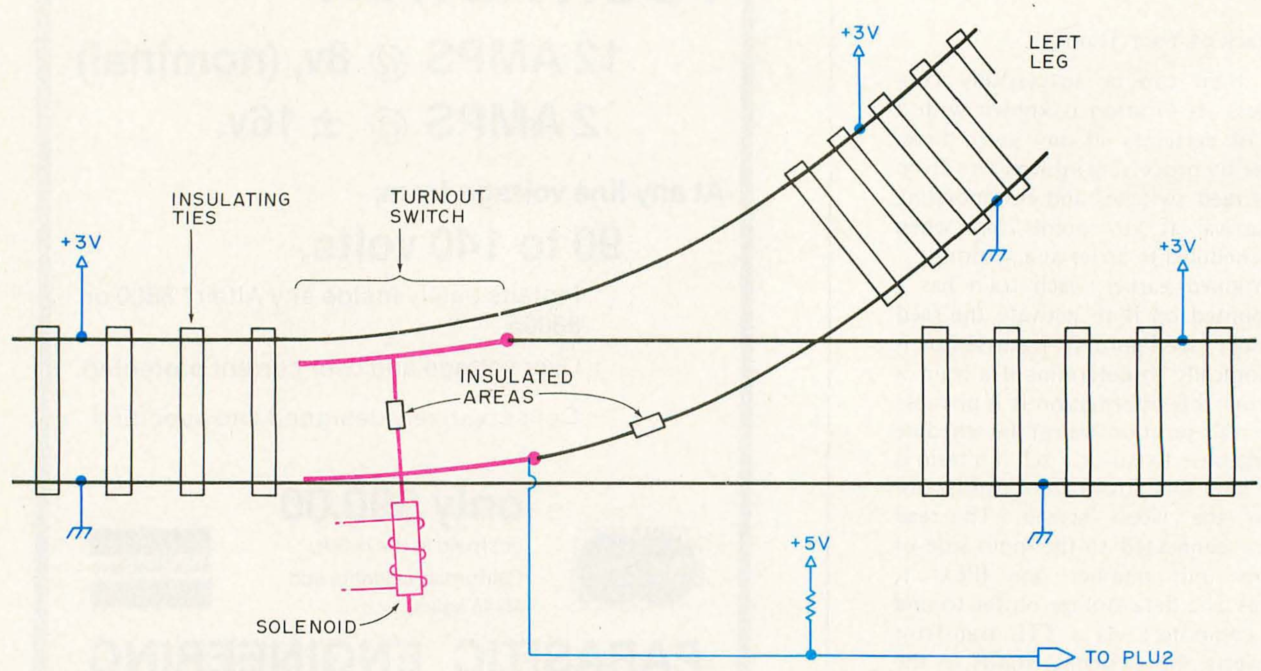


Figure 6: A typical fork (or "turnout") in the track. In this illustration, the turnout is positioned to divert the train to the left leg. These solenoid controlled mechanisms act as their own sensors. One leg of the switch is either grounded or ungrounded depending on its position. This is used to provide a logical 0 or 1 output to the computer for confirmation of the switch's position. The solenoid is illustrated in figure 7.

PLU 1. Each section of the track has been assigned a specific bit and only those bits of the word necessary to alter the status of the track section in question will change.

A Successful Turnout

Another aspect of control is the "turnout" switch (see figure 6). Turnout is a railroading term for a fork in the track which divides it into two branches. This term is used to avoid confusion with the electrical switches. The turnout settings will obviously have to change as a function of the two trains' positions and the programmed routes. The current status of a turnout must be determined by the computer before a train can pass through. A derailment will occur, for example, if the train proceeds from either leg of a turnout against the current setting. The turnout mechanisms in this system have been modified to act as their own sensors. One of the two moving legs of the turnout is electrically isolated in one position and grounded in the other. It is connected to a +5 VDC level (TTL) through a pullup resistor so that its output, which goes to parallel line unit number 2 (PLU 2), is either high or low depending on the switch's position.

A software routine controls the turnout mechanisms by indexing the track table in memory to determine which track switches must be thrown to get both trains to their next checkpoints. It then moves the turnouts to the required position, waits until the mechanical motion of the switches is com-

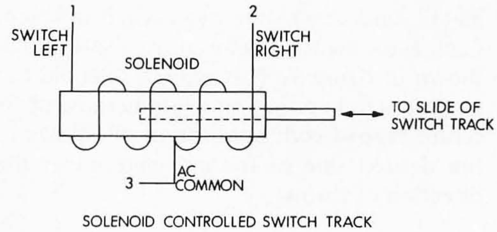


Figure 7: A turnout solenoid. This type of solenoid can be activated in either direction because of its center tapped coil. Application of voltage to the desired side of the coil determines the direction of throw.

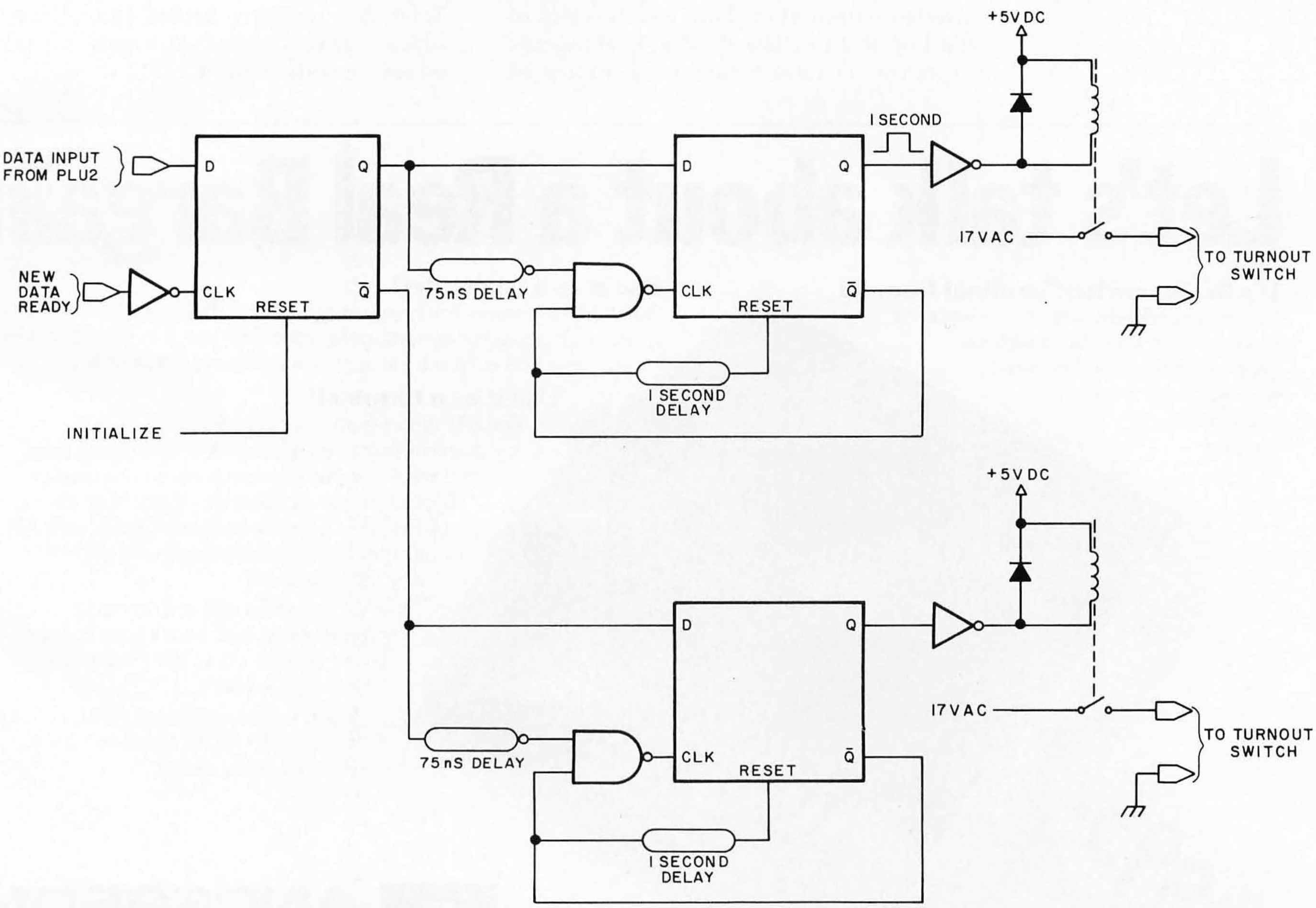


Figure 8: Coil activation circuitry. The coils used to energize the track solenoids cannot be energized 100% of the time (a 100% duty cycle) without burning up. This circuit uses flip flops and delay elements to provide a one second pulse to a relay where it is converted to 17 VAC, sufficient to activate the solenoid. This illustration, like the rest in this article, is intended to illustrate a concept rather than to serve as a basis for actual construction.

pleted, and checks the new switch position. Each track switch is moved by a solenoid as shown in figure 7. This type of solenoid can be activated in either direction because of its center tapped coil: application of voltage to the desired side of the coil determines the direction of throw.

Mortal Coils

Activating each solenoid requires a fairly elaborate arrangement, since PLU 2 has only 16 bits in its output word, while 32 control signals are actually required for all left and right solenoid positions. Further, the solenoids cannot merely use a level converter to increase the output control signal of PLU 2: such solenoids cannot be constantly energized at the required level without burning up. Figure 8 shows a circuit which converts the incoming signal to a level of 17 VAC used to energize the solenoid. The initial signal is sent to IC1, a hardware latch with both inverting and noninveerting outputs. This latch is used to buffer the incoming data. A 75 ns delay is used between the inverted output of the latch and the clock of flip flop IC2 to allow the data level to settle to its correct value before being clocked in. A logical 1 signal from PLU 2 will cause IC2 to go high and activate the coil. Another delay circuit from the inverted output of IC2 feeds back to the clear line and turns the relay coil off after one second. All of this causes the turnout switch to move to its right hand position. Shifting the switch to the left involves the same type of procedure. To insure the correct initial conditions, a software routine in the LSI-11 places all turnout switches in the left hand position following startup.

Off the Beaten Track

An emergency (eg: a train derailment) will occasionally occur which requires an immediate halt to all operations. A provision on the RT02 data terminal keyboard permits "panic stops" in such circumstances. To bring a more normal, orderly halt to the system, an appropriate command is entered into the data terminal: this sends train A to station 1 and train B to station 4. When train A reaches its station, power is reversed on the tracks so that it will back into siding 1. Train B is similarly backed into siding 2, which signals the end of a very tractable system demonstration!■



The TV Oscilloscope

Or, Some Notes on Building a Display

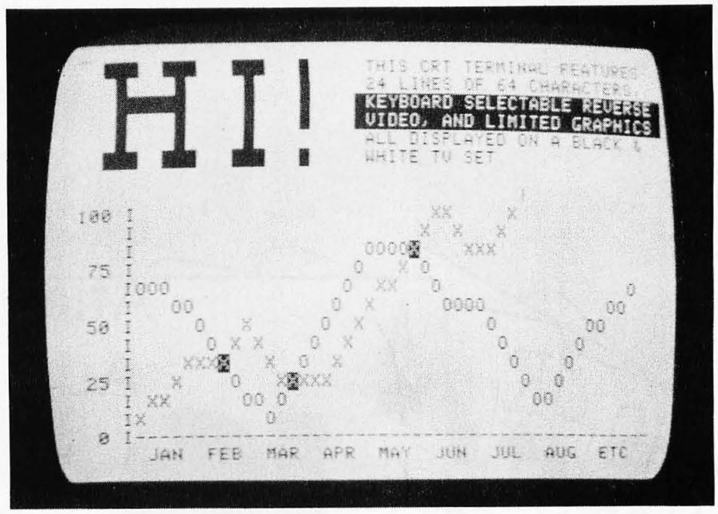


Photo 1: A "typical" display. This shows the completed display in operation. The large block letters are reverse video spaces. The use of blanking prevents generation of any signals at the top or bottom of the screen and keeps the visible portion of the display within the linear region of the TV set.

Kenneth Barbier
POB 1042
Socorro NM 87801

Although conceived and designed independently, the display circuitry in my CRT terminal is virtually identical to that of CW Gantt in his article "Build a Television Display" on page 16 of June 1976 BYTE. The major difference is that I did not use the MM5320 sync generator, and my display is therefore not interlaced and its horizontal sweep is not exactly at the usual TV horizontal rate. My display, shown in operation in photo 1, consists of 24 lines of 64 characters, with a frame consisting of 270 raster lines repeated 60 times per second. With this formulation I have a time allotment for 27 character lines with ten raster scan lines per character. The extra three character line positions represent overhead for vertical sync.

In a TV set the interlacing is produced not by the hardware within the set, but by the nature of the incoming sync signal. By using a decode of what would have been character line 26 (raster lines 251 thru 260) as the vertical sync, and a decode of character column 72 as the horizontal sync, the generator produces a horizontal sync frequency of 16,200 Hz. The TV set is happy with these small deviations from its usual sync diet, requiring no readjustment when used for its original purpose.

Another difference between the two approaches is in the size of the display. I wanted no fewer than 64 characters per line in order to be compatible with existing software written for a Teletype terminal. And to be consistent with more characters per line, using more lines per frame preserves the character's aspect ratio: the ratio of its height to width.

With 64 character spaces each seven dots wide, plus sync and retrace time allowances of about 20%, I found my dot rate would be

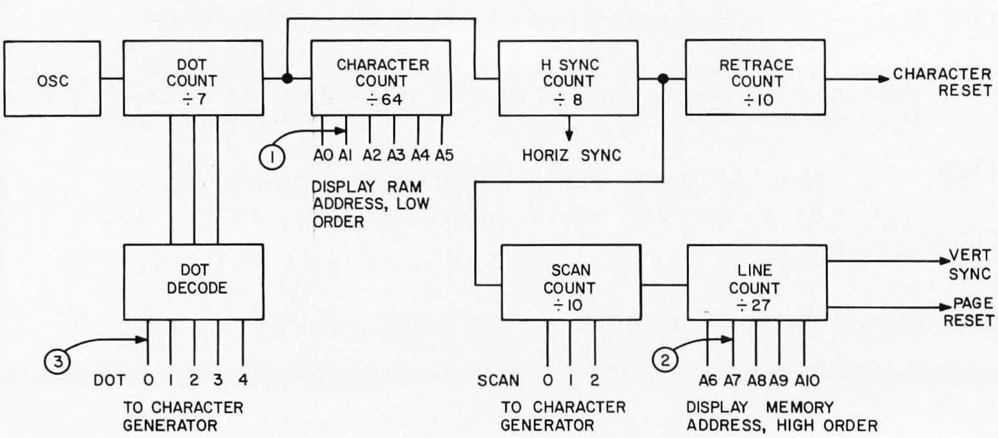


Figure 1: Block diagram of a television display. This version is very similar to the display described by C W Gantt in June 1976 BYTE, page 18. The points indicated by circled numbers are used as described in the text to generate the various test patterns shown in photos 2, 3, 4 and 6.

something over 9 MHz. How to push all that through a commercial TV set front end? No way! I never even tried. So I had to invent my own TV set interface which was described in my short article in July 1976 BYTE, page 38.

A simplified block diagram of the timing portion of my terminal is shown in figure 1. Since it varies only slightly from that of Gantt's, I have not included a complete detailed circuit diagram or explanation. At the block diagram level it can be seen that an oscillator drives a dot counter and decoder to produce a scan of the five dots making up each line of a character, followed by two dots worth of spacing. The seventh dot is used to increment the character counter, which counts the 64 characters to be displayed in each line. The end of the line is a count corresponding to what would have been character 72; this is used to increment the scan counter. The 65th thru 72nd character positions are timing overhead allowing time for horizontal sync.

The scan counter counts ten scan lines: one blank line above the characters, the seven raster lines comprising the characters, a blank line below the characters, and a line on which an underline or cursor position marker can appear. At the end of the tenth scan line, the character line counter advances. There are 24 displayable character lines numbered 0 thru 23. Line number 24 is for spacing at the bottom of the frame, line number 27 is reserved for the vertical sync and retrace time, and line number 26 is a blank line at the top of the screen. (Lines are numbered 0 to 26 consistent with the 27 binary states of the line counter.)

The outputs of the character counter (0 thru 64) and the line counter (0 thru 23) are used to address the display's programmable memory of 2048 bytes on two boards of 1024 bytes (1 K bytes) each. Only 1536 bytes of this memory are used in the display; the remainder is available to the processor for normal programming purposes. The processor can both write into the programmable memory area and read data from this memory. However, these operations should take place during vertical retrace time to avoid visible glitches. The processor needs to access the display memory in order to move the cursor around without disturbing the stored characters, and in order to implement scrolling when writing more than 24 lines. Being able to read and write the display memory makes it possible for these functions to be implemented entirely in software.

Enabling the processor as well as the display to read the memory also makes the hardware more complex than Gantt's. His approach was to implement a display only, mine was to build a smart video terminal. Both are built of the same foundation blocks: a sync generator, a memory address counter, and a character generator.

Whatever approach you might take, once you have built a composite sync generator and interfaced it to a TV set you have a powerful tool for developing the rest of the system, the main theme of this article. By breaking the video line shown on Gantt's

About the Author

Ken Barbier is an electronics technician employed by National Radio Astronomy Observatory, Socorro NM. In support of the construction of the largest radio telescope in the world, he has been designing and building microcomputer controlled automatic test equipment for the project's Digital Control System. This gives him the opportunity to divide his time between hardware design and programming. His other hobbies include building a microprocessor controlled bridge game, and flying, preferably without an engine.

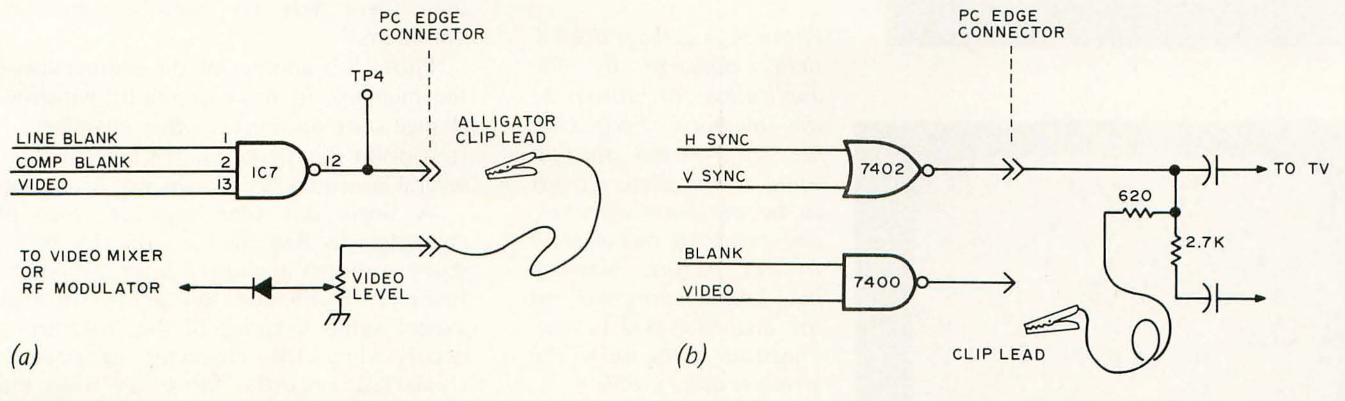


Figure 2: (a) A modification of the Gantt televison display circuit, of page 18, June 1976 BYTE, to use this test instrument technique. The line from IC7 pin 12 to the video level potentiometer is broken so that input to the video mixer, and thus to the TV, is taken from some arbitrary point under test rather than the usual source. (b) The author's video output circuitry, with a similar clip lead jumper normally in place for display purposes, but free to be moved around for testing purposes.

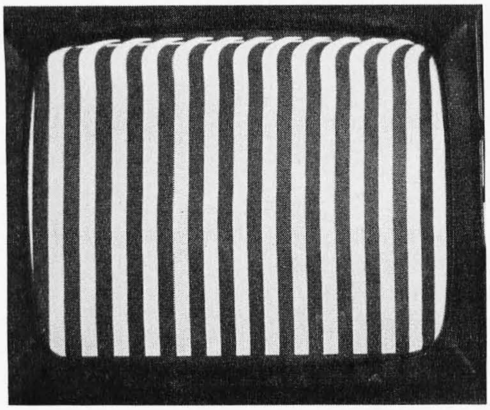


Photo 2: Exploring the circuit with the probing alligator clip produces various interesting signals. Here the clip looks at address bit 1, point 1 on figure 1. This signal changes state every other character position, giving vertical bars two character widths wide. Note the distortion at the top of the image, not normally seen due to blanking of the display.

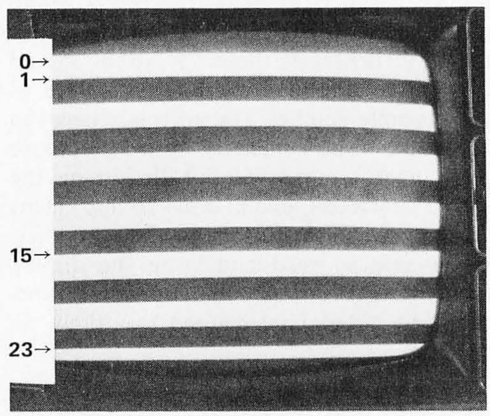


Photo 3: Moving the alligator clip probe to address bit 7 at point 2 in figure 1 yields this display. Here the line counts of several lines are shown at the left of the picture.

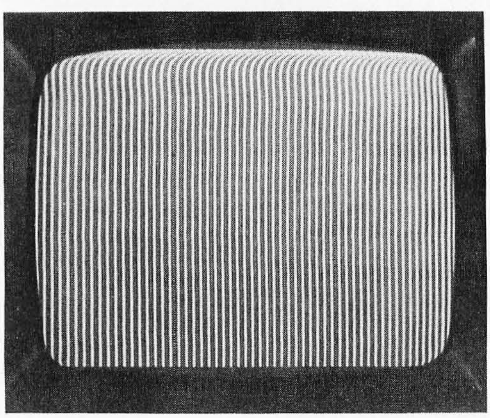


Photo 4: Looking at a single dot position. When the alligator clip is moved to point 3 on figure 1, this pattern is produced. Each character has seven horizontal dot positions, of which two are inactive spacing positions, and five are used for character generation.

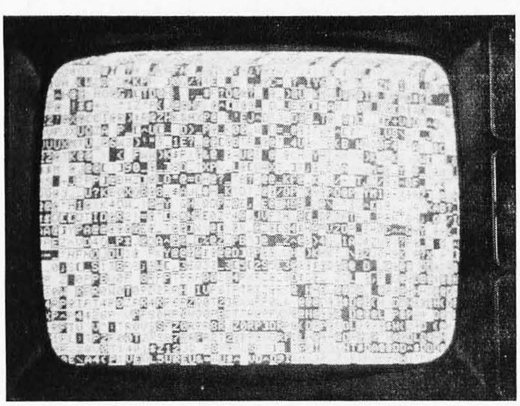


Photo 5: A garbage display field, obtained by the application of power to the memory. Each chip has a preferred start-up value, so this pattern tends to be the same whenever one turns on the display. In this picture, blanking has been suppressed, so the distortion due to nonlinearities at the top of the screen is quite visible.

schematic (June 1976 BYTE, page 18) between the output of IC 7 pin 12 and the level control and bringing it off the printed circuit board through the edge connector, we can insert an alligator clip lead in series as seen in figure 2a. The equivalent connection modifying my own interface, published in July 1976 BYTE, page 38, is shown in figure 2b. Now by using the clip lead as a test probe, any TTL level signal within the system can be displayed on the TV set to facilitate system development and debugging.

This TV oscilloscope will display signals as intensity modulation within the display field area, instead of as the vertical deflection on a real oscilloscope. Since the hardware we are building operates in synchronism with the field produced by the composite sync, and since our signals are all digital with a TTL logical 0 showing as white and logical 1 as black, this is not a real disadvantage.

For example, probing around in my system I can display the output of the next to least significant bit of the address to the character storage memory as seen in photo 2, test point 1 on figure 1. This output should change state for every other character position in the line. Counting the number of bars in this picture, I saw that not all of the 64 characters were going to be visible, since there are less than 32 bars showing in this display. This told me that some adjustments had to be made in the relationship between the character counter, the horizontal sync, and the time allowed for retrace. Also, as seen at the top of the raster in photo 2, the horizontal sync is imperfect following the vertical retrace. This came about in my system since the widths of the sync and equalization pulses were not made the same as TV standards. Since none of the 24 lines to be displayed are affected, I concluded that there was no need to cure this "problem."

Photo 3 is another of the address lines to the memory, in this case the bit which will change state once every other character line (test point 2 in figure 1). The line counts for several positions are shown on this picture.

A single dot time (one of seven per character) as displayed on the TV oscilloscope is shown in photo 4 from test point 3, figure 1. Examining this photo we might expect some smearing of the characters to occur when the character generator is connected, since the "dots" are wider than they are high. This is a result of the TV set trying to display a 9 MHz dot rate.

Up to this point we have been able to use the TV oscilloscope to test the portions of

Circle 15 on inquiry card.

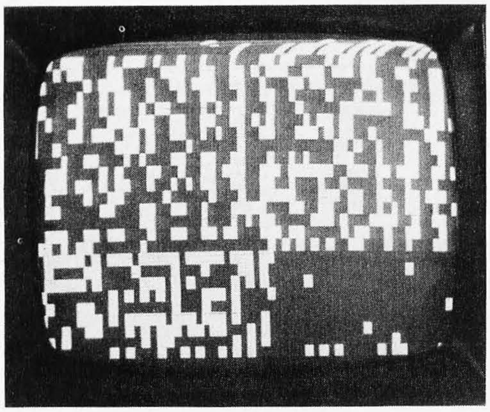


Photo 6: A map of the preferred states of one memory chip. By tapping into just one of the memory data lines with the display probe, it is possible to generate a map of the data. This data is identical to that of photo 5, but we are now looking directly at one bit, instead of generating characters from 8 bit byte groupings of several chips. This same technique of looking at one bit can be done with normal programmed displays, too, rather than the turn on pattern used here.

the system that scan the dots making up the characters and the scanning of the memory addresses. When all is well to this point, we could connect the programmable memory address lines to the data inputs of the character generator and display one of each character, as Gantt did on page 17 of June 1976 BYTE.

The next step is the connection of the display memory to the address lines, with the display memory data going into the character generator. Turning on the system with the probe on the character generator output, we find that the memory contents will initially be an arbitrary pattern, producing a screen full of garbage as in photo 5. Note how the horizontal "pulling" at the top of the screen distorts the characters in the top line. As this is actually line 26, it is not used in the display, and will disappear after the "blank" signal is connected as in figure 2b.

The arbitrary pattern is not totally so, since a virtually identical pattern will appear with each turn on. This is due to built-in asymmetries in the memory chips, with each bit having a preferred state to come up in when power is initially applied. We can connect our clip lead to one bit of the memory data as in photo 6 and see that each memory chip in the 2 K by 8 bit array has a unique pattern visible in these preferred states.

In photo 6, notice the obvious division in the pattern about two thirds of the way down the display. This corresponds to the division between the lower 1 K (lines 0 thru 15) and the upper 1 K (lines 16 thru 23) of memory, which use different chips.

After we connect the processor to the input side of the memory, we can use this same TV oscilloscope connection to examine the setting and resetting of each bit at each address under program control. Setting the memory to all zeros, each bit we turn on will appear as one black block in the display. The TV oscilloscope can thus be used to insure that the processor and display address



Photo 7: Physical layout of the equipment. The author's display is shown here as it was built on prototyping cards with a bus wired backplane between connectors. The clip lead is shown here attached to memory address bit 7, as used to generate photo 3. The computer and keyboard are in the foreground.

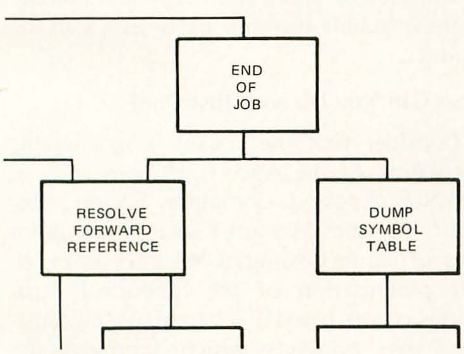
correspond correctly, and that data is written correctly. (Of course, we could just as well have put some message on the display, too.)

Photo 7 shows the hardware in my system. The clip lead can be seen connected to the display memory address line out of the timing board which was used to produce the display in photo 3. Since all of the dot signals, scan line signals, memory addresses, and the memory read and write data buses all appear on the backplane wiring of the system, virtually everything in the system is accessible to the TV oscilloscope test instrument.

For even more versatility in the TV oscilloscope, it might be useful to add an AND gate and another clip lead, so that the exact timing relationships between two signals could be determined. Alternately, one of the signals could be added into the video at a different level to produce a gray image so that the two signals could be differentiated. *[For looking at programmed behavior, the same technique can still be used if the vertical or horizontal sync signal is used as a timing cue. Much more could obviously be written on this subject as readers adapt Ken's idea to personal use . . .* **CH/**■

BYTE's Bugs

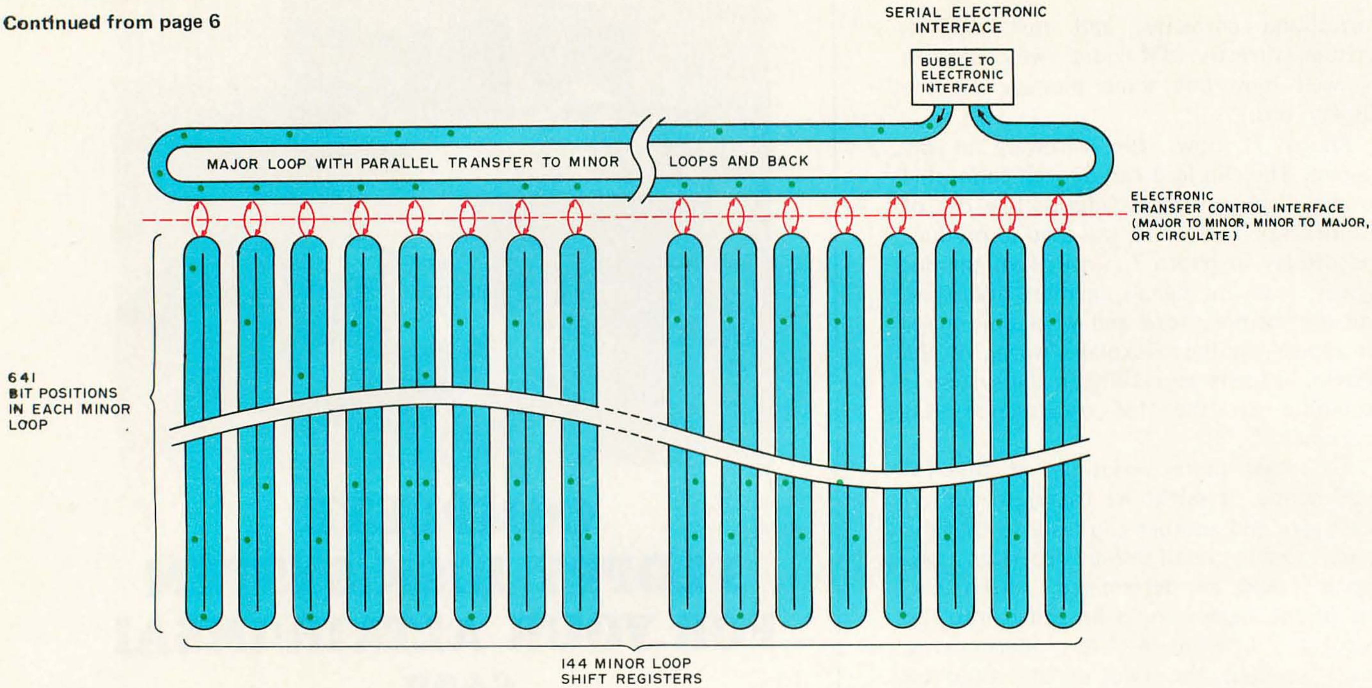
A Structure Glitch



Jack Emmerichs called upon his receipt of April 1977 BYTE and reported a couple of errors in the execution of figure 3 on page 66. (As Murphy's law would have it, figure 3 was moved to the first part of the article at the last moment at Jack's request, and he did not get to see author proofs for that one figure.)

The major glitch is corrected by noting the above segment of the diagram in its proper form. Two minor errors are an extraneous "level 3" at the lower left of the diagram, and an extraneous "Compare Strings" box in the "General Service Routines" shown at the lower right of the diagram.■

Continued from page 6



Write Operations:

1. Serially load major loop
2. Align minor loops to desired record
3. Transfer major loop to minor loops' top bit position

Read Operations:

1. Align minor loops to desired record
2. Transfer minor loop's top bit position to major loop
3. Serially unload major loop and use data

Figure 1: Conceptual diagram of a magnetic bubble film memory. This diagram is concocted from a general knowledge of the way bubble memories work, with captions based on the figures given by Texas Instruments' press release on the TBM0103 memory part. The memories to be used must also have support hardware (or software) to sequence the various steps needed to randomly or serially access desired 144 bit blocks stored in the chip's minor loop shift registers.

consists of serially loading the 144 bits into the major loop, then transferring all 144 bits into the minor loops; reading is done in reverse order by copying the currently positioned bits of all 144 minor loops into the major loop then serially reading the major loop. The result is a nonvolatile file storage device which has 641 blocks of 18 bytes per block, a total of 11,538 bytes in one 14 pin dual in line package measuring 1.0 by 1.1 by 0.4 inches (2.5 by 2.8 by 1.0 cm). Using a volume parameter for memory density, this elephant of a memory stores 200,000 bits per cubic inch (13,000 bits per cubic centimeter) in a nonvolatile magnetic medium.

With this short introduction to the existence of a new low cost, high density memory system, what are the ramifications of such a part for the personal computing user? I'll confine my comments to two major classes of application for the memory, applications which take advantage of its nonvolatility, low cost of .22 cents per bit in sample quantities, and the large size of each part's memory. The first class of applications for the bubble memory is as nonvolatile file storage in minimum size personal computing systems analogous to pocket calculators; the second class of application is as on line file storage for larger general purpose desk top or console personal computers. Superficially, the difference between the two classes of application is in the number of these bubble memory parts used with the product.

What Can You Do with Just One?

Consider first the pocket programmable calculator. At the April 6 1977 meeting of the New England Computer Society, Bev Pettit of the Hewlett-Packard calculator sales office in Lexington MA gave an excellent presentation of the functional capabilities of the new HP-67 and HP-97 calculators, two products which represent the ultimate in pocket calculator performance to date. These machines can be used to perform fairly sophisticated problems, with overlay structures for data and program material swapped using the magnetic card medium incorporated in the design. But, the machines are limited by the fact that there is a finite set of data registers (26, if I recall correctly) and 224 programming steps. What would be the impact of adding a bulk program storage facility implemented with the TI bubble memories or equivalent? Viewed only as a calculator, with 8 bit command codes for each function, just one such bub-

ble memory chip would lead to a device which could permanently store interpretive calculator programs with an aggregate total of 11,538 program steps, or an improvement of program capacity nearly two decimal orders of magnitude. Or viewed as data memory, the same single chip could accomodate over 1400 floating point numbers in an 8 byte representation. (I have no information on the details of the HP-67's internal representation, but assuming thirteen 4 bit BCD mantissa digits, two BCD exponent digits, and one 4 bit field for signs, an 8 byte floating point representation is a fair estimate for the typical calculator's characteristics.) But simply to generalize the calculator's programming techniques to a much larger program and data storage field is not the ideal situation. For example, the HP-67 and HP-97 systems use an interpretive "label" operation code to identify places within a program. When the label is referenced by a "go to" command or "call subroutine" command, the internal operation is to search linearly through the program memory field looking for a label operation code followed by the correct identification. This works quite acceptably in a calculator with a 224 step capacity, but would produce unacceptably long delays in a memory field of the size represented by just one of these new bubble memory chips.

As the new capacity gets incorporated into the small personal computers called programmable calculators, we can expect to see the beginnings of more sophisticated calculator operating systems and interpreters with features needed to take advantage of such memory. The lowly hand held calculator will start having the data management features of the magnetic disk based minicomputer, but miniaturized to numeric symbols and smaller total capacity. Perhaps we might even see a limited (ie: somewhat inconvenient for touch typists like myself) alphanumeric data storage mode for program names and variable symbols in the next generation of calculators, reminiscent of the "Minisec" of Arthur C Clarke's *Imperial Earth* which I summarized in my April 1977 editorial.

What Can You Do with Just "N"?

In a more expensive console or tabletop personal computing system, the same characteristics of nonvolatile electronically controlled bulk storage which expand the calculator concept can be applied to the general purpose computer system concept. Basically, the bubble memory makes possible a very large totally electronic on line buffer storage memory, at a price compatible with personal computing economics. Since the memory is

Articles Policy

BYTE is continually seeking quality manuscripts written by individuals who are applying personal systems, or who have knowledge which will prove useful to our readers. Manuscripts should have double spaced typewritten texts with wide margins. Numbering sequences should be maintained separately for figures, tables, photos and listings. Figures and tables should be provided on separate sheets of paper. Photos of technical subjects should be taken with uniform lighting, sharp focus and should be supplied in the form of clear glossy black and white or color prints (if you do not have access to quality photography, items to be photographed can be shipped to us in many cases). Computer listings should be supplied using the darkest ribbons possible on new (not recycled) blank white computer forms or bond paper. Where possible, we would like authors to supply a short statement about their background and experience.

Articles which are accepted are typically acknowledged with a binder check 4 to 8 weeks after receipt. Honorariums for articles are based upon the technical quality and suitability for BYTE's readership and are typically \$25 to \$50 per typeset magazine page. We recommend that authors record their name and address information redundantly on materials submitted, and that a return envelope with postage be supplied in the event the article is not accepted. ■

totally electronic there is no problem with mechanical design considerations, an advantage which is offset by the constraint of not having removable media as is the case for a floppy disk. As the price of the parts goes down with eventual mass production, the cost per installed bit has the potential of dropping well below that of the removable media floppy disk storage devices.

The implications of this bubble memory for the "appliance" computer are most interesting. Let's consider what the new "minimum" personal computing system which incorporates the bubble memory medium might look like. First, the system includes a typical microprocessor design with eight of the contemporary 16 K dynamic memory parts for main memory, with perhaps 8 K to 16 K of systems software in read only memory. Peripherals built-in include video display (interface or built-in monitor depending on the manufacturer), and alphanumeric keyboard, with a high speed audio cassette interface for long term program storage. For filing of current programs and data this minimum system incorporates eight of the TBM0103 bubble memory units, giving a capacity of 92,304 bytes of data or program file capacity. With an appropriate operating system and high level language built into the read only memories, the symbolically named files in this bubble memory region can be rolled in or out of the 16 K volatile programmable memory region as desired. Using the audio tape interface at the end of a programming session, the file memory could be checkpointed automatically onto tape while the user attends to some other noncomputer related activity like eating, sleeping, etc. (Assuming an audio tape interface at 100 bytes per second, it would take 923 seconds or about 15 minutes to dump the state of the file memory, plus about three minutes to dump the programmable volatile memory of 16 K bytes.)

What would such a conception cost? Perhaps \$2000 to \$3000, assuming production quantities of the bubble memory cost significantly less than \$200, that the support circuitry is not excessive, and that the rest of the system uses existing technology. This would be the minimum configuration of such an "appliance computer circa 1977-1978." The logical next improvement in performance would be to incorporate one Shugart style minifloppy disk file, at perhaps a \$500 to \$800 increment in price. The bubble film memory is functionally

very similar to a floppy disk. The bubble memory's data transfer rate, at 50 kbps, is comparable to the transfer rates of the small floppies, and its access time to blocks of data is approximately one to two orders of magnitude faster (12.8 ms versus typically 250 to 500 ms head positioning time on a floppy disk). But the bubble memory chip capacity of 92,000 bits is about the same as the capacity of the typical 5 inch floppy disk. Using one floppy disk drive for its removable media attributes, and the standard permanently "mounted" simulated floppy disk of the bubble memory, this appliance computer concept now has the ability to "instantly" copy disk files, sort data within the constraints of two 92,000 byte file regions, and do many of the "memory intensive" tasks usually associated with large interactive systems.

When Will We Users See Products?

The announcement by Texas Instruments, accompanied by extensive advertisements in the electronics trade press for the bubble memory, a 64 K charge coupled device, volatile memory and large 16 K dynamic programmable random access memories, is aimed primarily at designers of new products. TI happens to be the first to commercially market the bubble memories. (AT&T has already been using them for several months in production phone equipment, but these parts are not available to general commercial users.) If we use the example of the Shugart minifloppy disk drives as a prototype for the delay from announcement to proven end user product for a system of similar complexity, then we can expect to see end user deliveries of mass storage subsystems for present day personal computer systems sometime late this coming fall. (This editorial was written April 8 1977.) As for complete computer systems which make the bubble memory an integral part, I would tend to expect the first such systems in the first half of 1978 or sooner depending upon the cleverness and resources of the companies which ultimately produce such products. In the meantime, perhaps some technologically enterprising reader will purchase the early sample versions of the bubble memory chips along with documentation, and design a homebrew computer interface which we can publish for the benefit of all BYTE readers. The bubble memory, a long awaited promise, has finally become a technological and commercial reality available for use in personal computing products.■

Continued from page 34

- Type 03: External Symbols

Byte number	Description
1	Dollar sign ($) delimiter.
2, 7	Up to 6 ASCII characters of the external symbol name. The name is left justified, blank filled.
8, 9	Record type 03.
10, 13	Last address which uses the external symbol. This is the start of a link list which is described below. The most significant byte is first.
14, 15	Binary checksum.
CRLF	Carriage return, line feed.

The Mostek SDB-80 assembler outputs the external symbol name and the last address in the program where the symbol is used. The data records which follow contain a link list pointing to all occurrences of that symbol. This is illustrated in figure 1.

1. The external symbol record shows the label (LAB) and the last location in the program which uses the symbol (212AH).
2. The object code at 212A has a pointer which shows where the previous reference to the external symbol occurred (200FH).
3. This backward reference list continues until a terminator ends the list. This terminator is FFFFH.

This method is easy to generate and decode. It has the advantage of reducing the number of bytes of object code needed to define all external references in a program. Intel defines each external reference explicitly.

- Type 04: Relocating Record

The addresses in the program which must be relocated are explicitly defined in these records. Up to 16 addresses (64 ASCII characters) may be defined in each record.

Byte number	Description
1	Dollar sign ($) delimiter.
2, 3	Number of sets of 2 ASCII characters. Where two sets define an address.

4-7	ASCII zeros.
8, 9	04
10	Addresses which must be relocated.
Last 2 bytes	Binary checksum.
CRLF	Carriage return, line feed.

• Assembler Object Output Sequence

The SDB-80 Assembler will output object records in the following sequence:

1. Internal symbols (type 02) in alphabetical order.
2. External symbols (type 03) in alphabetical order.
3. Data records (type 00), interspersed with relocation records (type 04).
4. End of file record (type 01).

Internal symbol records will be outputted only if internal symbols are defined in the source program. The same is true of external symbol records. Relocation records always will be outputted by the assembler.■

GLOSSARY

Object module The output from an assembler is called the object module. One or more object modules can be placed into memory by a loader. The output from the loader is called a "load module."

Relocatable program: A program which has extra information in the object module is referred to as relocatable. The extra information allows the loader to place the program anywhere in memory. A nonrelocatable program can be loaded only in one place in memory in order to work properly.

Position independent program: A program which can be placed anywhere in memory is called position independent. It does not require relocating information in the object module.

Internal symbol: A symbol which is defined in a program and whose location is made known to all other programs is called an internal symbol. It is also called "public" (Intel), "defined," "global," or "common."

External symbol: A symbol which is used in a program but which is not defined in the program is called an external symbol. Such symbols must be defined as internal symbols in other programs.

Linkable program: A program with extra information in the object module which defines internal and external symbols is called a linkable program. The loader uses this information to connect, or link, external references to internal symbols.

Speech Recognition for a Personal Computer System

James R Boddie
Bell Laboratories
Murray Hill NJ 07974

The August 1976 issue of BYTE presented to the personal computer user an introduction to half of the problem of man-machine communication by speech: machine generation of speech. Speech synthesis has been investigated since at least the 1800s. High quality speech can be produced by a computer if it decodes utterances that have been efficiently coded and stored in memory. Poorer quality, but sometimes acceptable speech can be synthesized by electronically simulating the human speech producing mechanism.

Speech communication in the other direction, machine recognition of speech, is usually considered to be a harder task. The purpose of this article is to present some of the problems and to introduce some solutions that may be realized by the personal computer enthusiast.

The ideal speech recognizer would be a machine that could accept casually spoken sentences from many speakers with different dialects and transcribe them in real time into written text or translate them into commands for controlling some function. Unfortunately, there are at least five major problem areas in speech recognition that are still incompletely solved.

- **The Continuous Speech Problem.** In ordinary conversational speech our words tend to "run together" acoustically. Say "How are you" and "How (pause) are (pause) you." The continuous version is uttered almost as a single word. What is the best procedure for segmenting such an utterance into words?
- **The Multiple Speaker Problem.** People say the same words differently. Can a machine allow for these differences or must it be "tuned" to each individual user?
- **The Limited Vocabulary Problem.** Error rates, recognition rates and memory requirements increase with vocabulary size. How many words can be reliably handled by a machine?
- **The Vocabulary Entry Problem.** The words in the vocabulary have to be described to the computer in some way. What is the most efficient way?
- **The Noise Problem.** Speech is filtered by some transmission process or corrupted by noise in transmission media such as telephone or radio. What is the best signal processing technique for these conditions?

These are the problems that concern many industrial and academic research laboratories using the most sophisticated techniques and equipment available. However, there are some realistic goals for a speech recognizer in a personal computer system. That is, it should be possible to build a system which can accept isolated words from a limited vocabulary of ten to 20 words spoken by a speaker whom the machine has been trained to understand. These goals are well within the capabilities of most contemporary microprocessors.

Several such systems have already been designed using a modest amount of discrete

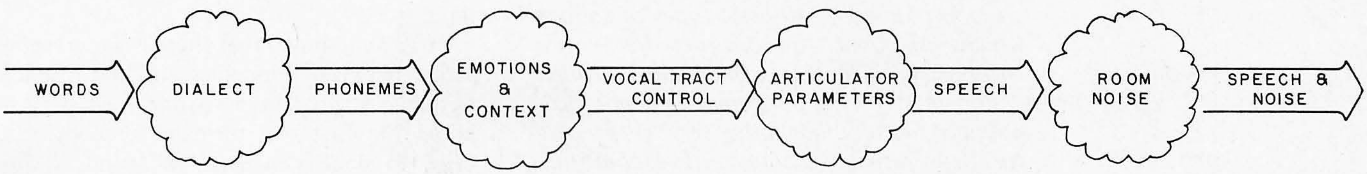


Figure 1: A speech encoding model. Messages from the brain in the form of desired words are processed through a series of transformations to produce the finished speech. These messages come in the form of "phonemes," or word building blocks (see table 1), which make up the desired word. This code is translated into appropriate neuromuscular signals which control the speech "articulators": the vocal cords, vocal tract, lips, tongue and teeth. The other parameters in the figure show how dialect, emotions, context and external noises can all have an effect on the final speech.

logic. (See the first six references in the bibliography at the end of this article.) Each of these systems can be easily duplicated with a small amount of hardware such as a microphone, amplifier, filters, an analog to digital converter and a microcomputer. This paper will show how one of these well known recognizers can be implemented. But first, a few basic facts about human speech production.

The Speech Code

In order to construct a system for decoding speech you should know how the acoustic signal is encoded with the desired message. One way of looking at the process is shown in figure 1. The message from the brain may be thought of as going through a series of transformations which modify it by many factors. The word idea or linguistic information is first translated into speech sound units called phonemes. This "code" is a collection of the descriptors of the sounds necessary to say the words. In English there are approximately 40 basic types of sounds which are used to construct our entire vocabulary. The phonemic code is translated by the central nervous system into neuromuscular signals for controlling the physical speech making apparatus or articulators. The articulators "modulate" the sound generated by the vocal cords and produce the radiated acoustic signal.

This process has sources of error. The phonemic codes for various dialects are different from one another. A speaker may say the same word many different ways depending on context or physical and emotional state. Differences in the sizes of people's articulators account for variations between speakers. Day to day changes in the articulators (due to a head cold, for example) can cause variations in the way a given person sounds. Finally, the acoustic signal itself may be corrupted by noise or filtering.

The message coding process produces a signal which contains much information that is not significant to the intended message and which makes any direct comparison of speech waveforms difficult if not impossible. The extent of the coding inefficiency can be seen by a simple example. The following sentence:

> "The objective of this section is to develop these techniques."

contains 60 characters and can be spoken in about three seconds. Using a 5 bit code for alphabetic characters, real time transmission at speech rates would require only 100 bps. If the utterance were transmitted with a string of 42 phonemes coded with six bits each, then 84 bps would be necessary. However, if the speech signal itself were band-limited at 3 kHz, sampled at 6 kHz, and quantized to 256 levels or eight bits, real time transmissions would require a rate of 48,000 bps. If a computer could analyze this digitized speech in real time it would have to process one 8 bit speech sample every 160 μs. A microcomputer with a cycle time of 500 ns would have only 320 cycles between speech samples. That does not allow time for very many instructions to analyze and match the data with reference waveforms. The direct storage of the waveform would require 6,000 bytes of memory for every second of speech. It should be clear that it is desirable to have a method of rapidly extracting the useful information, thereby reducing the amount of data storage and computation time. A look at the speech production mechanism provides some clues for data analysis and reduction.

Speech Parameters

Speech sounds originate from two sources. So-called "voiced" sounds are made by the vocal cords and make up vowel-like sounds. The vocal cords produce periodic bursts of air through the vocal tract at a repetition rate of about 125 to 200 Hz.

"Unvoiced" sounds or "fricatives" are

Voiced	Example
AE	bad
AH	father
AI	bite
AW	bought
AY	bay
EH	bet
EE	beet
ER	bird
IX	fit
OU	boast
UX	book
UH	but
UU	boot
WX	win
YX	yes
RX	rip
LX	lit
MX	man
NX	not
NG	ring
Fricatives	
FX	fan
TH	bath
SX	sip
SH	ship
CH	church
HX	hand
Combinations of Voiced and Fricative Sounds	
VX	van
DH	than
ZX	zip
ZH	measure
Stops	
JH	jump
PX	pan
TX	tan
KX	can
Voiced Stops	
BX	ban
DX	dan
GX	gab

Table 1: Speech phonemes. In English there are approximately 40 basic types of sounds which are used to construct the entire vocabulary. These sounds are called "phonemes." The phonemes are not pronounced literally, but rather serve as symbols for their respective sounds. The phonemic code is translated by the central nervous system into neuromuscular signals for controlling the physical speech making apparatus or "articulators." "Voiced" phonemes are the vowel-like sounds which are produced in the vocal cords as periodic bursts of air from 125 to 200 times a second. "Unvoiced" phonemes or "fricatives" are noise-like sounds made by forcing air through constrictions in the mouth. Some phonemes are combinations of voiced and unvoiced sounds. "Stops" result from momentary blockage of air flow (and sound).

noise-like and are made by forcing air through constructions in the mouth. If the air flow is completely blocked for a moment, the sound is called a "stop." Sometimes we make both voiced and fricative sounds. Table 1 classifies various phonemes into these source types. Phonemes have special symbols that are often used, but this table uses a machine readable format.

The vocal tract is like a pipe from the vocal cords to the lips which has several natural resonant frequencies called "formants." This pipe acts like a filter that most readily passes sounds whose frequencies are the same as the formants. The three most prominent formants are centered around 500, 1500 and 2500 Hz but they vary with changes in position of the tongue, lips and mouth. The voiced and unvoiced sounds can be thought of as "carrier" signals that are "modulated" by the resonances in the vocal tract.

It is well known that there is a correspondence between phonemes and the formant frequencies. In fact, it is only necessary to know the first two formant frequencies in order to identify many vowel sounds. Figure 2 shows a map of several vowel sounds in a plane formed by the first two formants. For example, if you knew that for some interval of a speech utterance the first formant was at 800 Hz and the second at 2500 Hz, then you might guess that the speech sound was the phoneme "EH."

A nice property of formant data is that it changes rather slowly. It is necessary to sample the formant frequencies only every 10 to 20 ms in order to keep up with changing phonemes. This means that the motion of two formants could be stored in only 100 bytes per second of speech as opposed to 6000 bytes per second for the waveform itself.

Naturally, we would like a technique for easily measuring the first two formant frequencies by looking at the waveform data. The most accurate methods require either a lot of high speed computation or special purpose hardware. Fortunately there is a very simple way to get an estimate of the formants which is most suitable for microcomputer processing.

Zero Crossing Analysis

Since the late 1940s it has been known that the amplitude of the speech waveform carries little information. Most of the useful information can be obtained from just the knowledge of the number of times the signal waveform crosses the zero axis. This is called zero crossing analysis. *[This method would lend itself perfectly to Walsh transform analysis. . .***CM***]* If you count the number of times the signal changes sign in a fixed interval, say every 10 to 20 ms, you can get numbers which closely follow the changes in the first formant frequency. If you first emphasize the higher frequencies by high pass filtering, then the zero crossing rate tracks the second formant frequency.

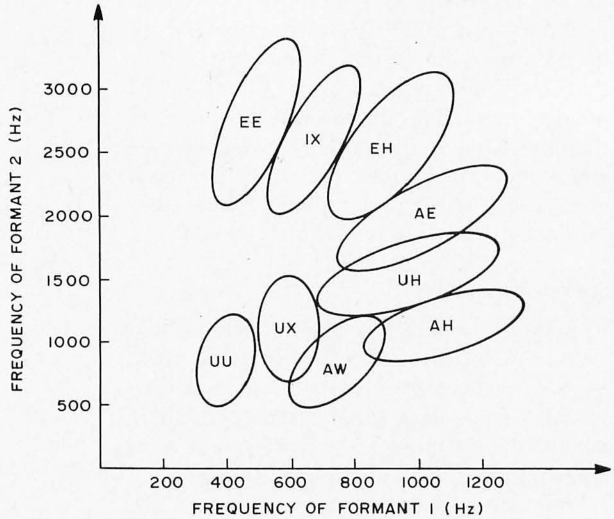


Figure 2: Vowel phonemes on the formant plane. "Formants" are the natural resonant frequencies of the vocal tract. These resonances are similar to the resonant vibrations of organ pipes or the characteristic sound of an empty bottle when air is blown across it. The three most prominent formants are centered around 500, 1500 and 2500 Hz, but they vary with changes in position of the tongue, lips and mouth.

Although the zero crossing rates do not give accurate estimates of formant frequencies, they do contain enough information to partially separate the vowels for recognition purposes. Figure 3 shows a map of vowels on a high pass versus low pass zero crossing rate plane.

There are many advantages in using zero crossing analysis techniques in digital speech recognition. One of them is that the necessary hardware is simple. Since only the sign of the waveform is important, only one bit per sample is needed, which eliminates a requirement for an analog to digital converter in the system. The analysis can easily be done in real time since no multiplications are involved – only counting. The zero crossing parameters can be measured with simple hardware or software as the signal comes in, which eliminates the necessity of storing the waveform. Because the amplitude information is not used, the analysis is independent of voice volume and inflection. This makes zero crossing measurements somewhat less dependent upon characteristics of individual speakers.

Pattern Matching

After the acquisition of the zero crossing parameters, the identification of the word can be done using one of two simple processes: in a one step approach, the parameter data is directly compared with reference patterns (sometimes called templates) for each word in the vocabulary. The closest match is chosen as the most probable utterance. In a two step method, the parameter data is first converted into a string of phonemes. Then this string is compared with a phonetic representation of the words in the vocabulary.

Both methods have problems. The one step comparison suffers if the word is spoken at a different rate from the words in the sample set unless some technique is used to find the best time alignment. The time

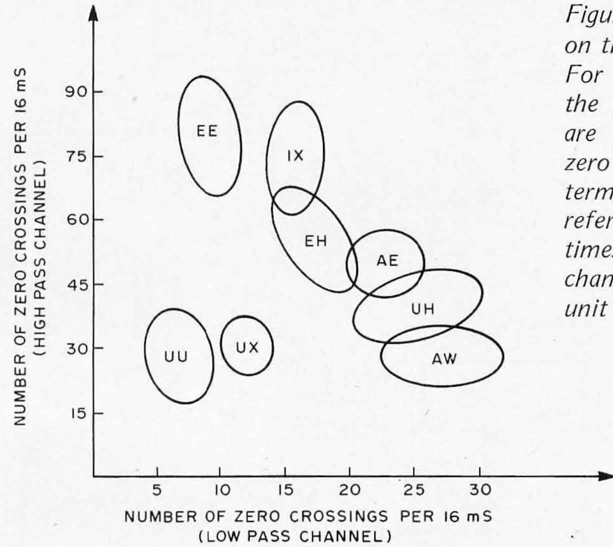


Figure 3: Vowel formants on the zero crossing plane. For comparison purposes the formants in figure 1 are plotted here on the zero crossing plane. The term "zero crossing rate" refers to the number of times that the voice signal changes its polarity per unit time (see figure 4a).

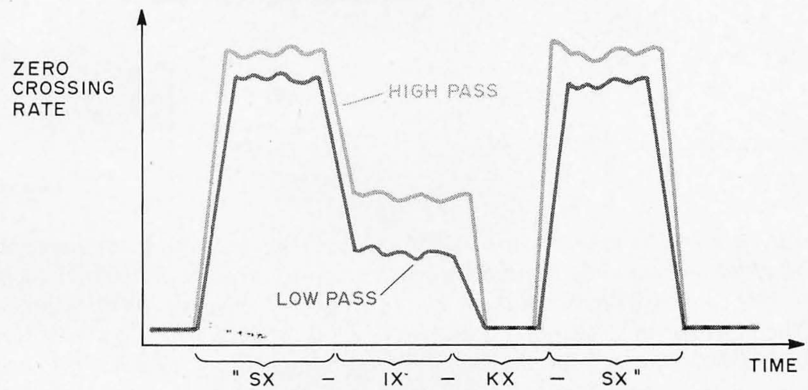
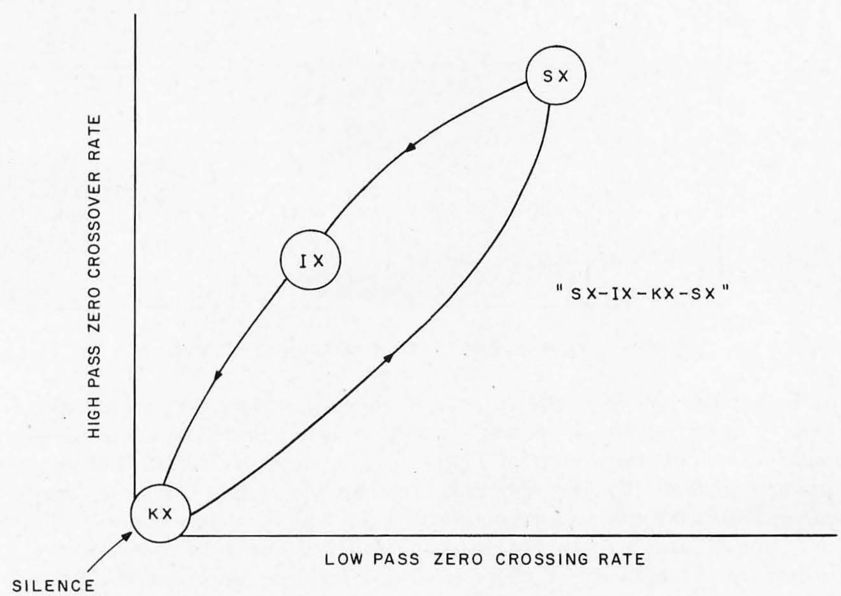


Figure 4a: A plot of the word "SIX" showing the zero crossing rate as a function of time. The word is shown split into its four phonemes. In this case, the signal is first processed through two bandpass filters. The number of zero crossings is then determined by the computer. Results for both filters are shown. As the graph indicates, there is a very short period of silence during the pronunciation of the KX phoneme which can be verified by slowly pronouncing the word "six."

Figure 4b: A diagrammatic representation of the word "SIX" on the zero crossing plane using a form of state diagram. In this graph the high and low pass functions from figure 4a are plotted against one another in stylized fashion to better show how the zero crossing rates change as the word is pronounced. (Note that the word begins and ends with the same phoneme, SX.) The signal is split into two bandpass regions because these particular regions reveal key attributes of English words which can act as types of "fingerprints" of the words for later indentification.



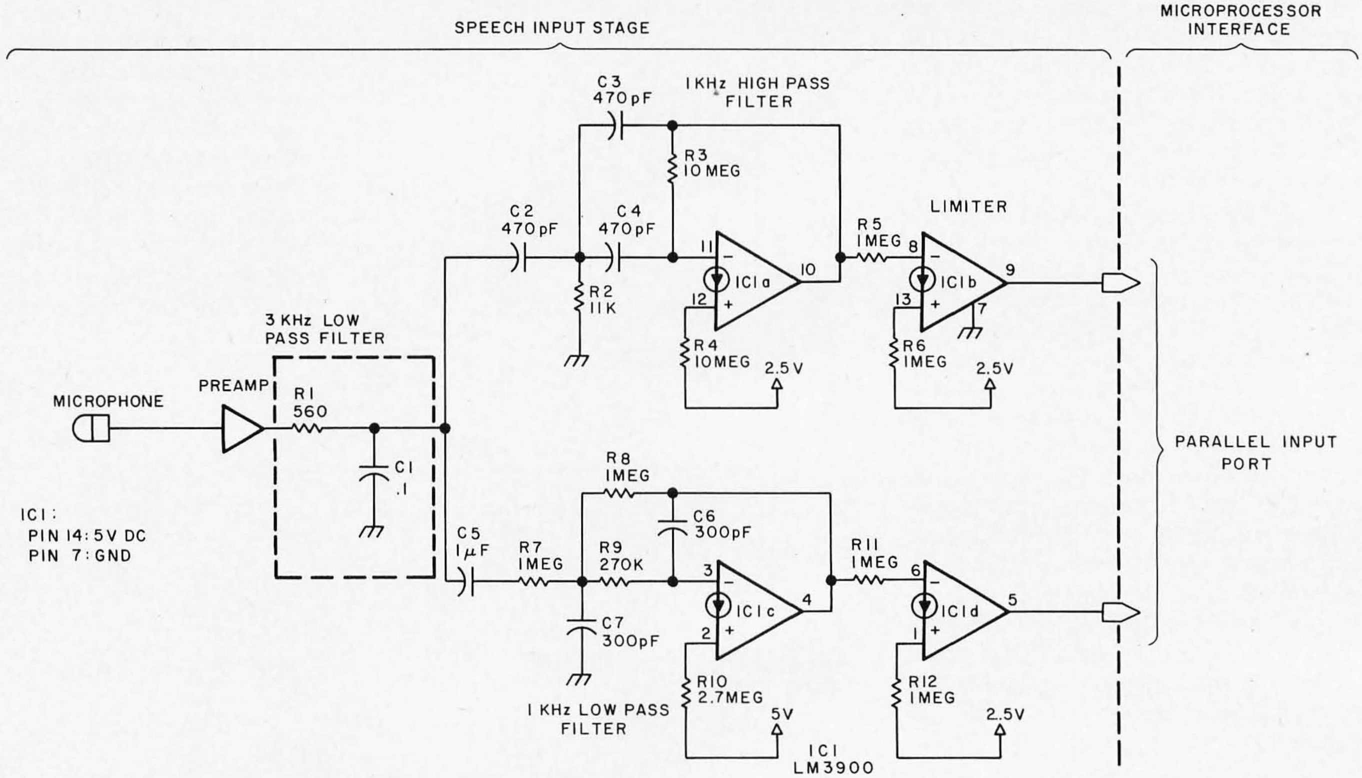


Figure 5: Speech filtering circuitry. A voice signal is picked up by the microphone and preamplified. It is then sent through a low pass filter made up of C1 and R1 whose cutoff frequency is 3 kHz. The signal is next passed through a 1 kHz high pass filter and a 1 kHz low pass filter as shown. The two signals then go to the microprocessor interface for further processing. IC1, which forms the basis for this circuit, is a quad LM 3900 op amp. The diode-like symbols shown on the four amplifiers indicate the use of "current mirrors" for the noninverting inputs. For a detailed discussion see National Semiconductor's book, Linear Integrated Circuits, *page 2-250 and following.*

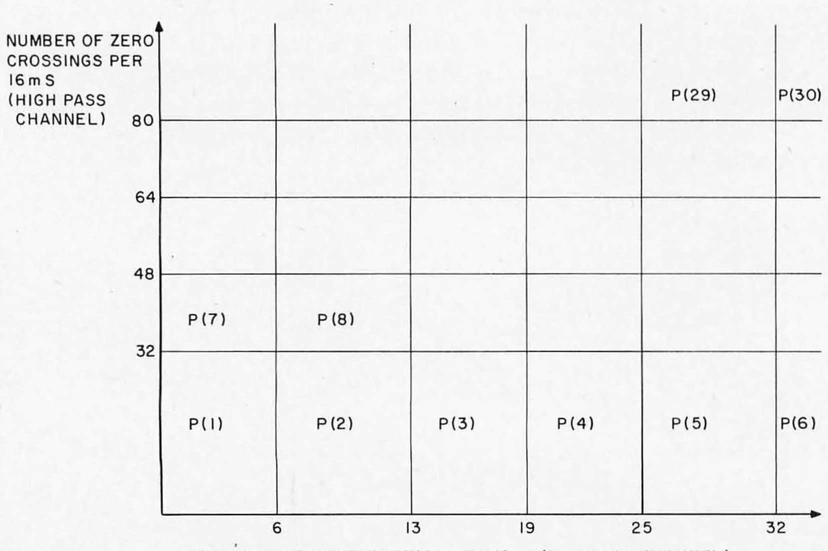


Figure 6: In order to implement the speech recognition algorithm, 60 samples of speech over roughly 1 second's duration are sequentially processed through the circuit illustrated in figure 5. For each of the 60 intervals the number of high and low pass zero crossings per second is counted and stored in a software array corresponding to the regions in this figure. (This figure is not to scale.) After this, a straightforward statistical correlation is performed on the array to compare it with a series of word models in the memory and to arrive at the most likely match.

dependence can be eliminated by considering the pattern of the data plotted on a zero crossing plane. Figure 4 shows how the word "six" might be plotted. The one step method has the double disadvantage that a lot of memory is required for template storage and that the matching may be more time consuming. Two step systems use less memory in the template storage but depend on the performance of an imperfect phoneme indentifier. The phoneme matcher must be able to tolerate errors such as missing, mislabeled or extraneous phonemes in the hypothesized string. For example, the output from a phoneme indentifier given in the utterance "six" (or "SX IX KS SX") might be "SX SX EE IX IX EH KX SX SX."

A Practical Speech Recognizer

You have probably realized by now that there are many possible approaches to the analysis and recognition of speech. With this in mind, we can now consider in detail an isolated word recognizer that can be implemented on most microcomputer systems. This algorithm is based on one of the earliest successful word recognition ex-

```
FOR J = 1 TO 60
F1(J) = 0
F2 (J) = 0
FOR I = 1 TO 100
   :
[Delay for 160 μS loop time]
   :
INPUT X                                              ;Input speech port
X1 = XOR (X, X0)                                     ;Exclusive OR input with last sample
IF X1 > 1 THEN F2 (J) = F2 (J) +1                    ;State change or bit 1
IF (X1 = 1) OR (X1 = 3) THEN F1 (J) = F1 (J) +1      ;State change or bit 0
X0 = X
NEXT I                                               ;Save last sample
NEXT J
```

Listing 1: The input algorithm. Arrays F1 and F2 store the number of zero crossing counts in each 16 ms interval for the low pass and high pass channels respectively. This listing, like the following listings, is written in BASIC for illustrative purposes, but it must be translated into machine language to achieve real time performance.

```
      FOR I = 1 TO 30
      P (I) = 0
      NEXT I
      FOR J = 1 TO 60
      FOR I1 = 1 TO 5
      IF G1 (I1) > F1 (J) THEN 100
      NEXT I1
      I1 = 6
100   FOR I2 = 1 TO 4
      IF G2 (I2) > F2 (J) THEN 200
      NEXT I2
      I2 = 5
200   I = (I2–1)*6+I1
      P(I) = P (I) +1
      NEXT J
```

Listing 2: The classification and accumulation routine. This routine takes the information from arrays F1 and F2 and stores it in array P, which represents the zero crossing plane.

```
FOR J = 1 TO L                          ;Correlate
R (J) = 0
FOR I = 1 TO 30
R (J) = R (J) + P (I)*T (I, J)
NEXT I
NEXT J
J1 = 1                                  ;Pick best
R1 = R (1)
FOR J = 2 TO L
IF R (J) > R1 THEN R1 = R (J) : J1 = J
NEXT J
PRINT J1
```

Listing 3: The correlation routine. This routine compares the completed array from listing 2 with various word models in memory and uses standard statistical methods to find a best match. For a ten word vocabulary the entire speech recognition algorithm can be run in less than 69 ms.

```
INPUT "TEMPLATE NUMBER," L
FOR I = 1 TO 30
P1 (I) = 0
NEXT I
FOR K = 1 TO 8
INPUT "READY?", Q
   :
[Call routine for sampling speech and computing P]
   :
FOR I = 1 TO 30
P1 (I) = P1 (I) + P (I)
NEXT I
NEXT K
FOR I = 1 TO 30
T (I, L) = P1 (I)/8
NEXT I
```

Listing 4: The template generation routine. A "template" is a word model in the form of an array. This routine, which can be run off line (in BASIC if desired, since there are no time restrictions), is used to generate all of the word models in the vocabulary desired. Eight samples of a given word are uttered by the operator and averaged together to create each word template.

periments (see reference 1). Originally built with mainly analog components, it could recognize telephone quality digits from a single speaker with an accuracy better than 97 percent in real time. A microcomputer using a minimum amount of extra hardware can easily duplicate this performance.

The input stage is shown in figure 5. The speech is low pass filtered at 3 kHz and split into a 1 kHz high pass channel and a 1 kHz low pass channel. These filtered signals are compared with a reference voltage level and logic outputs indicate the result of the comparison. That is, a logic 0 results from a signal level greater than the reference and a logic 1 results from a level less than the reference. Proper adjustment of the reference levels will allow the measurement of the sign of the signal. The outputs from this stage are connected to the two least significant bits of a parallel input port.

The microcomputer must sample the input port every 160 μs and count the number of state changes (zero crossings) over 16 μs intervals. If the input is sampled for 60 intervals, then the machine will have "listened" for 0.96 seconds, which is more than enough time to pronounce most individual words.

Listing 1 is the input algorithm. It is written in BASIC for illustrative purposes, but it must be efficiently translated into machine language to achieve real time performance. All variables are 1 byte integers unless otherwise noted. The inner loop must be executed once every 160 μs. This can be done by using software delays or an external clock. The listing does not show details of how this delay is accomplished. The F1 and F2 arrays are used to store the zero crossing counts in each 16 ms interval for the low pass and high pass channels respectively. Of course, the "IF" statements can be replaced by a simple "rotate into carry, branch if carry" in the machine language code.

The next step is to divide the low pass versus high pass (F1, F2) plane into 30 regions as shown in figure 6. Then for each of the 60 intervals, the zero crossing counts in F1 and F2 are located on this plane and associated with one of the regions. A 30 element array P keeps up with the number of points in each region. This distribution of points in the F1, F2 plane is used as the input pattern to be compared with the reference patterns.

Listing 2 illustrates the classification and accumulation routine. The G1 and G2 arrays contain the information about the division of the F1, F2 plane. They should be initialized (in BASIC or equivalent machine code) as follows:

```
READ G1
DATA 6, 13, 19, 25, 32
READ G2
DATA 32, 48, 64, 80
```

The calculation of index I involves a multiplication by a constant 6. This can be most quickly done by coding:

```
I = (I2-1)*2 + (I2-1)*4 + I1
```

This BASIC statement can easily be translated into machine language using two left shifts and addition operations.

The pattern matching is done by correlating the P array with precomputed reference patterns for each word in the vocabulary. The word with the best match is picked as the most likely utterance.

Listing 3 gives the pattern matcher routine, expressed in BASIC. L is the number of words in the vocabulary. R is an array that contains the calculated correlation values for each word. T is a 30 by L array of reference templates. The number corresponding to the template which best matches the input is printed as J1. The calculation of R requires 30 times L multiplications, which can be very time consuming. For a ten word vocabulary and an 8080 processor (one 8 by 8 bit software multiply in 230 μs) the R calculation can be done in 69 ms. Also note that while the numbers in P are always positive, this does not apply to the numbers in T. R should be a double precision (2 byte) number to avoid overflow.

The templates are generated by averaging several sample patterns for each word. Much of this task can be done off line rather than in real time, so it can be coded in BASIC or other interpretive languages. A direct machine language coding is relatively straightforward, however.

Listing 4 shows how eight patterns are averaged to form a template. The program uses the code of listings 1 and 2 to obtain the pattern in array P. The accumulation of the P1 array should be done with 2 byte arithmetic. The division by 8 is accomplished by a simple right shift three bit places.

Finally, a little data massaging is necessary. To make this type of correlation pattern matching work, each template should have the same statistical properties. Specifically, each template should have a mean value of zero and a normalized standard deviation. This is easily done with the program in listing 5. The removal of the mean is simplified by the fact that the mean for each template is the same (60 intervals divided by 30 regions gives a fixed mean value of 2). The normalization of the standard deviation involves computing a square root. If the square root function subroutine is not available, then it can be computed manually with a pocket calculator and entered by hand (PRINT S : INPUT S2). This is not too painful since it needs to be done only once for each word in the vocabulary. S should be a 2 byte integer, and the division by S2 should be done using double precision arithmetic.

```
FOR J = 1 TO L
FOR I = 1 TO 30           Subtract mean
T (I, J) = T (I, J)—2
NEXT I
S = 0
FOR I = 1 TO 30
S = S + T (I, J) * T (I, J)
NEXT I
S2 = SQRT (S)
FOR I = 1 TO 30           ;Normalize
T (I, J) = 8 * T (I, J)/S2
NEXT I
NEXT J
```

Listing 5: A normalizing routine. This auxiliary routine is used to adjust the statistical properties of the templates. Specifically, each template is modified so that it has a zero mean value and a normalized standard deviation. This is done to insure that all of the word templates have the same statistical properties prior to correlation.

The templates can be saved and used indefinitely without alteration, but new templates will be necessary for different speakers. Composite templates can be made by averaging the reference patterns of several talkers, but they do not work as well as individualized ones.

This very simple speech recognizer is only one of many systems that can be adapted to microcomputers. Much more information on the digital analysis and synthesis of speech can be found in the *IEEE Transactions on Acoustics, Speech and Signal Processing*, which is available in most engineering libraries.■

BIBLIOGRAPHY

1. Davis, Biddulph, and Balashek, "Automatic recognition of spoken digits," *Journal of the Acoustical Society of America*, volume 24, pages 637-642, November 1952.
2. Ross, "A limited-vocabulary adaptive speech recognition system," *Journal of the Audio Engineering Society*, volume 15, pages 414-416, October 1967.
3. Gilli and Meo, "Sequential system for recognizing spoken digits in real time," *Acustica*, volume 19, pages 38-44, 1967.
4. Teacher, Kellett, and Focht, "Experimental, limited vocabulary speech recognizer," *IEEE Transactions on Audio and Electroacoustics*, volume AU-15, pages 127-130, September 1967.
5. Purton, "Speech recognition using autocorrelation analysis," *IEEE Transactions on Audio and Electroacoustics*, volume AU-16, pages 235-239, June 1968.
6. Bezdel and Bridle, "Speech recognition using zero crossing measurements and sequence information," *Proceedings of the IEEE*, volume 116, pages 617-623, 1969.

An Application for This Speech Recognition System

An excellent application for Dr Boddie's system would be to use it as a quick way to enter hexadecimal code without toggling or typing. The vocabulary required could be kept to a minimum if the operator pronounces each digit separately (ie: "seven, seven," not "seventy seven") so that the various compound forms do not have to be learned. The computer would then verify the verbal entry by outputting the code to a printer or display screen. Better yet, use this system in conjunction with a computerized speech synthesizer such as D Lloyd Rice's Computalker [*BYTE, August 1976, page 16*] to obtain a verbal interaction. A more sophisticated system could process verbal op code mnemonics and assemble them directly in hexadecimal code on your display screen for verification.

Cassette tape and other magnetic tape recording without bit errors requires that we get our feet wet in the murky waters of error correcting codes . . .

How to Pick up a Dropped Bit

W Douglas Maurer
Rm 634, University Library Bldg
George Washington Univ
Washington DC 20052

The oxide coating of a tape is sometimes unevenly distributed, particularly when the tape is old and has been used many times.

The phenomenon of the dropped bit causes difficulties in two distinct areas of computer technology: in the recording of data on tape (or disk, or the like), and in the transmission of data from one place to another. Suppose, for example, that we are recording one hundred 32 bit words on a tape. Out of the 3200 bits that are to be written on the tape, there is a nonzero chance that at least one of them will be wrong. Either it will be recorded as a zero, when it should have been a one (a dropped bit) or it will be recorded as a one, when it should have been a zero (an added bit). Even if all 3200 bits are recorded correctly, there is still a nonzero chance that the next time we read this tape, we will read at least one of the one bits from the tape as if it were a zero, or one of the zero bits as if it were a one. In such a case we again speak of dropping a bit, or adding a bit. Often both dropped and added bits are referred to, generically, as dropped bits, and we shall continue to do so in this paper.

In a similar way, suppose we are transmitting a message which consists of bits. (The message does not have to involve computers at all; it may, for example, simply be a message from one Teletype to another.) In a long message there is, again, a very good chance that at least one bit which is transmitted will be received in the wrong way. It might be received as a zero, when it is supposed to be a one, or vice versa. Again we speak of bits being dropped in transmission. (One slightly confusing piece of terminology here is that the entire collection of bits, out of which a very few are dropped, is very often referred to as a "message," even when we are not transmitting it, but rather recording it on tape or disk.)

There are many possible sources of dropped bits. Tapes often have tiny dust particles on them which interfere with the reading and writing of data. The oxide coating of a tape is sometimes unevenly distributed, particularly when the tape is old and has been used many times. The same considerations, of course, apply to floppy disk memory, or any other kind of memory involving an oxide coating. In transmitting messages from one place to another, noise in the channel and receiver can very easily degrade the quality of the reception.

In order to solve the problems created by dropped bits, we can proceed in two general classes of ways. The first is to improve our hardware in such a way that dropped bits do not occur: We can clean our tapes. We can throw away our old tapes. We can transmit messages at a slow rate, and so on. The other approach is what may be called "picking up" the dropped bits. The idea is to send a message that is longer than the original one, and that is so designed that, even if certain bits are dropped, the original message can be recovered. (As we mentioned above, the word "message" is being used here in a general sense; it may, in particular, be a record written on tape or disk.)

Picking up a dropped bit is referred to, more precisely, as "error correction," a term which must be carefully distinguished from "error detection." In error detection, we simply detect the fact that some bit has been dropped; we cannot tell which bit is the

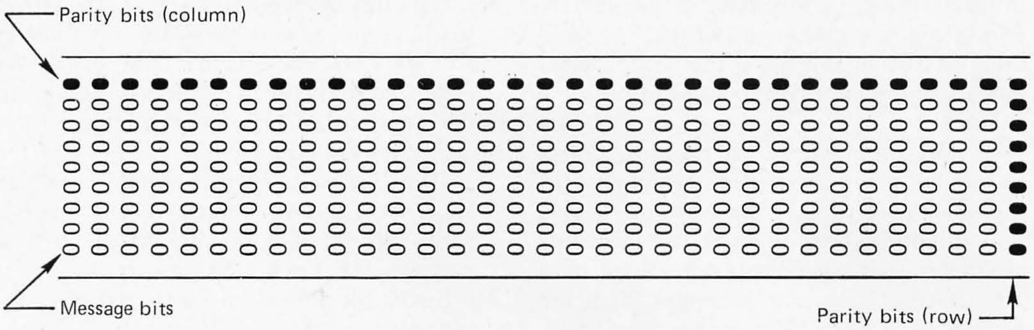


Figure 1: A block of data, eight bits plus parity in height, 32 bits plus parity in width, used as an example in the text. This layout of bytes might be thought of as a block of data on a standard 9 track tape drive; or it might be an internal memory image of data sent and received through a serial data port, bit by bit, as in the personal computer's audio tape interfaces.

wrong one. If we could tell that, then we could correct the error, because if a bit is "one," and it is wrong, then it must really be "zero," and vice versa. Knowing which bit is wrong is what enables us to do error correction.

Both error detection and error correction are affected by the question of how many bits in a message are wrong. If we can tell which bit is wrong, assuming that there is only one wrong bit, then we have *single error correction* – even though we might not be able to correct more than one error in a message (or we might correct these errors in the wrong way). In the same way, if we can tell, for example, that something is wrong whenever one or two (but no more than two) bits in a message are wrong, then we have *double error detection,* even though there might be some patterns of more than two wrong bits which are such that we cannot tell that anything is wrong.

A simple way to perform error detection is by means of what is commonly called parity checking. In large systems, 8 bit bytes recorded on an IBM standard tape are always accompanied by a ninth bit, the parity bit. This bit is so chosen that the total number of one bits among the nine bits (the original eight, plus the parity bit) is an odd number (1, 3, 5, 7 or 9). (Sometimes it is done the other way around, that is, to give even parity instead of odd parity; but we shall ignore this alternative for the moment.) Now let us suppose that no more than one of these nine bits has been dropped. When the bits are read again, if the total number of one bits is an even number, we know that there has been a dropped bit. We do not know, however, which of the eight bits was dropped. (Actually, for all we know, it might be the ninth bit – the parity bit itself – that was dropped.)

Parity checking is extended, on IBM standard tapes, to provide for the possibility of error correction as well as error detection. This is done by performing "two-dimensional" parity checking, as shown in figure 1. Each column has a parity bit, and each row also has a parity bit. Now suppose that exactly one of the bits in the message of figure 1 was dropped. If that bit was in (say) the fifteenth column and the second row, then the parity in the fifteenth column will be wrong – there will be an even number of one bits in it, rather than an odd number – and the parity in the second row will also be wrong. If these two parities are wrong, it is then a simple matter to find the bit in the fifteenth column and the second row and change it (from a zero to a one or from a one to a zero).

The assumption that is made in this parity scheme is that dropped bits will be infrequent enough that, in a message (or a record on tape) of this size, only one bit, at the most, will be dropped. For this reason the two-dimensional parity checking scheme of figure 1 is said to provide single error correction, but not double error correction. Suppose that the bit in the seventeenth column and the first row is also dropped. This means that the fifteenth column, the seventeenth column, the first row and the second row will all have wrong parity. In this case, which two bits are wrong? Let us denote by (x, y) the bit in column number x and row number y. In this case, it is bits (15, 2) and (17, 1) that were dropped. But it could just as easily have been bits (15, 1) and (17, 2), and exactly the same erroneous behavior would have occurred. In other words, we can't tell, when there is a double error, where the double error is if this scheme is used, and thus, we have no way of correcting it.

Both error detection and error correction are affected by the question of how many bits in a message are wrong.

The two-dimensional parity checking scheme does, however, provide double error detection. Whenever there is a double error, that is, whenever exactly two bits in the message are dropped, we can detect this fact. This is true even when both errors are in the same column. Of course, in that case, there will be no way to tell what column the errors are in. All of the column parities will be right, because an odd number of bits with two changes in it remains an odd number of bits. The only way we can tell that some-

thing is wrong, in this case, is that exactly two of the row parities are wrong. The same thing is true in reverse, if the two dropped bits are in the same row, but not in the same column.

In fact, this scheme will not only detect all double errors, it will detect almost all multiple bit errors. If there are more than two dropped bits, as long as these are randomly distributed (and not caused by a scratch on a tape, for example), then the probability is overwhelming that all of them, or at least most of them, will be in different columns, and thus there will be quite a number of column parities that will be wrong. We cannot refer to the given scheme as a multiple error detection scheme, in general, because there are some cases in which errors go completely undetected. For example, consider the four bits we treated earlier, namely bits (15, 2), (17, 1), (15, 1), and (17, 2). Suppose that these four bits are all dropped, and that no other bits are dropped. Now we have an undetectable error: All our parities, including those in the fifteenth and seventeenth columns and in the first and second rows, will be right. But this is so infrequent an occurrence that it may, for all practical purposes, be ignored.

Standard parity checking schemes of this kind can be improved upon in two ways: The first is by increasing their efficiency; the second is by increasing the number of errors that can be corrected. We shall treat these points one at a time.

Suppose that the record in figure 1 contained eight rows and 32 columns, for a total of 256 bits. If we include the check bits there are nine rows and 33 columns. This means that there are 9+33 = 42 different check bits. (The row of check bits has to have a check bit of its own, of course, and so does the column of check bits. Sometimes these are the same, but even if they are, there are still 41 check bits.) In contrast, we will now exhibit a clever scheme that requires only eight check bits. It performs single error correction, just as does the scheme of figure 1. It has, however, certain disadvantages which we will discuss later.

The scheme is as follows. We number the bits in our message from 0 to 255. The last of the eight check bits will be a parity bit for half of the bits in the message, namely the bits numbered 1, 3, 5 and so on up to 255. Note that these are the bits such that the bit number (1, 3, 5 and so on), when it is itself expressed in binary, as an 8 bit quantity, has a one bit in the last position (indicating that it is an odd number).

The next to last of the eight check bits will again be a parity bit for half the bits in the message. This time, however, it will be for the bits numbered 2, 3, 6, 7, 10, 11 and so on, up to 254 and 255. These bit numbers are all such that, if they are themselves expressed in binary, as 8 bit quantities, then the next to last bit of each of these 8 bit quantities will be a one bit.

The general scheme should now be apparent. Each of the eight check bits is a parity bit for half the bits in the original message. For $1 \leqslant k \leqslant 8$, the k-th check bit is a parity bit for all the bits that have the following property: If the bit number is N, and if N is expressed as an 8 bit binary quantity, then the k-th bit of this quantity is a one bit. In particular, the first of the eight check bits is a parity bit for bits 128, 129, 130 and so on, up to 255, of the original message.

Suppose now that one of our bits is dropped. For definiteness, let us suppose that it is the 99th bit. We express the number 99 as an 8 bit binary quantity: 01100011. And now let us look at our eight check bits. Which ones of them are going to be wrong? The last one will be wrong, because 99 is an odd number, and therefore the 99th bit (which was dropped) is one of the 128 bits (half of the original 256) of which the parity was taken to form this last check bit. The next to last check bit will also be wrong, because 99 has the property that the second bit from the right in its binary representation is a one bit. The first check bit, though, will still be right, because this represents parity on the 128th, 129th, 130th, etc, bits, and none of these bits were dropped.

The general pattern should now be apparent. If we look at the eight check bits from left to right, and if we write a zero for each parity check that was right, and a one for each parity check that was wrong, we obtain the pattern 01100011. This is exactly the number 99 expressed in binary. And this means that we can tell that it was, in fact, the 99th bit that was dropped – which, in turn, means that we can correct the error. Thus we have a single error correction scheme, just as before, enabling us to pick up one dropped bit.

There are three problems with this scheme. The first is as follows. Suppose that there are no errors at all in a given 256 bits. Then, of course, all the check bits will be right, and we will obtain the pattern 00000000. But this pattern tells us that bit number 0 is wrong! In fact, none of our check bits involve parity on bit number 0 at all, and thus we have no way of telling whether this bit was dropped or not. This problem, however, can be solved rather simply. We transmit only 255 bits of data, instead of 256, and these are numbered from

Standard parity checking schemes of this kind can be improved upon in two ways: The first is by increasing their efficiency; the second is by increasing the number of errors that can be corrected.

Suppose that there are no errors at all in a given 256 bits.

1 to 255. The efficiency of the scheme is not too badly affected; we now have eight check bits for every 255 data bits, rather than eight for every 256.

The second problem with the scheme is that we have not considered the possibility that one of the check bits might be wrong. Suppose that the last check bit is wrong, and that all the other bits are right. Then using the scheme above, we would obtain the 8 bit pattern 00000001, and this would tell us that it was bit number 1 of the original 256 that was wrong. In general, an error in any of the bits numbered 1, 2, 4, 8, 16, 32, 64 or 128 can be confused with an error in one of the check bits. The solution, however, is again very simple: We just leave these bits out. We are now transmitting only 255 − 8 = 247 bits of data, with eight check bits, and again the efficiency is not too badly affected. If the 8 bit pattern we obtain contains seven 0 bits, and only one 1 bit, then we know that it is a check bit that is wrong.

The third problem, however, is more serious. Suppose that more than one bit out of the 256 was dropped. Note that in our scheme, if there is an error of any kind, we determine one particular bit position to change. In other words, we always assume that if there is an error, it is a single error. If there is a multiple error we are always going to do the wrong thing. This is in contrast with the two-dimensional parity checking scheme, in which we almost always know that something is wrong, no matter how many bits get dropped. The only solution to this problem is a partial one: We can forget about correcting errors and use this scheme to detect errors only; and if we do this, all double errors will be detected. In other words, this scheme can be used for single error correction or double error detection, but not both.

A remarkable property of our scheme is that the eight check bits can all be generated simultaneously. We take the exclusive OR of all the binary integers 00000001 thru 11111111 (or 1 thru 255 in decimal) — leaving out 1, 2, 4, 8, 16, 32, 64 and 128, as noted above — which correspond to one bits in the message. That is, if the **i**-th bit in the message is a one bit, then the integer **i**, written in binary, is exclusive ORed with all other integers **i** with the same property. The resulting 8 bit quantity consists of precisely the eight check bits we need. For $1 \leqslant k \leqslant 8$, the **k**-th bit of this quantity is the exclusive OR of as many one bits as there are positions **i** in the message, such that bit **i** is a one bit and the **k**-th bit of the integer **i** is also a one bit, together with a number of zero bits, which do not affect the exclusive OR. An algorithm for performing this process is as follows:

1. Initialize so as to point to the first bit of the message.
2. Set R1 = 3. (R1 will contain the index **i** as above.)
3. Set R2 = 0. (R2 will contain the eight check bits.)
4. Set R3 = 4. (R3 will be 4, 8, 16, 32, etc, as above.)
5. If the current bit in the message is a zero bit, skip the next step (that is, go to step 7).
6. Set R2 equal to the exclusive OR of R2 and R1.
7. Point to the next bit of the message.
8. Set R1 = R1 + 1.
9. If R1 ≠ R3 then go to step 5.
10. Set R3 = R3 + R3.
11. If R3 ≠ 512 then go to step 7.

At this point, if we are writing a message, we append the eight check bits in R2 on to the end of the message. If we are reading a message, we read the next eight bits and form the exclusive OR of these bits with R2. If the result is zero, the message is without error. If it is 1, 2, 4, 8, 16, 32, 64 or 128, then one of the check bits is wrong. If it is anything else — call it **i** — then the **i**-th bit is wrong, and must be changed (to a zero if it is a one, or vice versa).

It should also be clear that there is a scheme like this for any number **m** of check bits. We have here taken $m = 8$, and the number of data bits is $2^m - m - 1 = 256-8-1 = 247$; but we could have taken $m = 4$, for example, obtaining four check bits for each 11 bits of data. This provides another approach to the problem of multiple errors in 256 bits; we can require only that there be no multiple errors in 11 bits (say), at the cost of a certain loss of efficiency.

Is it possible to pick up more than one dropped bit at a time? That is, can we devise a scheme that is capable of double error correction? Yes, we can; we can even provide **n**-tuple error correction, for any (fixed) positive integer **n**. Schemes for doing this, however, are quite complex, and their complexity increases with the number of errors to be corrected. There is a whole subfield of electrical engineering called the theory of error correcting codes, which concerns itself with schemes of this kind. It is remarkable that error correcting codes involve one of the few known practical applications of the theory of Galois fields. (Every mathematician knows the tragic story of Galois, a French math student back in the Age of Dueling who got involved in a challenge to a duel, and, knowing his opponent was a far better duelist than he, spent his last night on earth

Suppose that more than one bit out of the 256 was dropped.

Is it possible to pick up more than one dropped bit at a time?

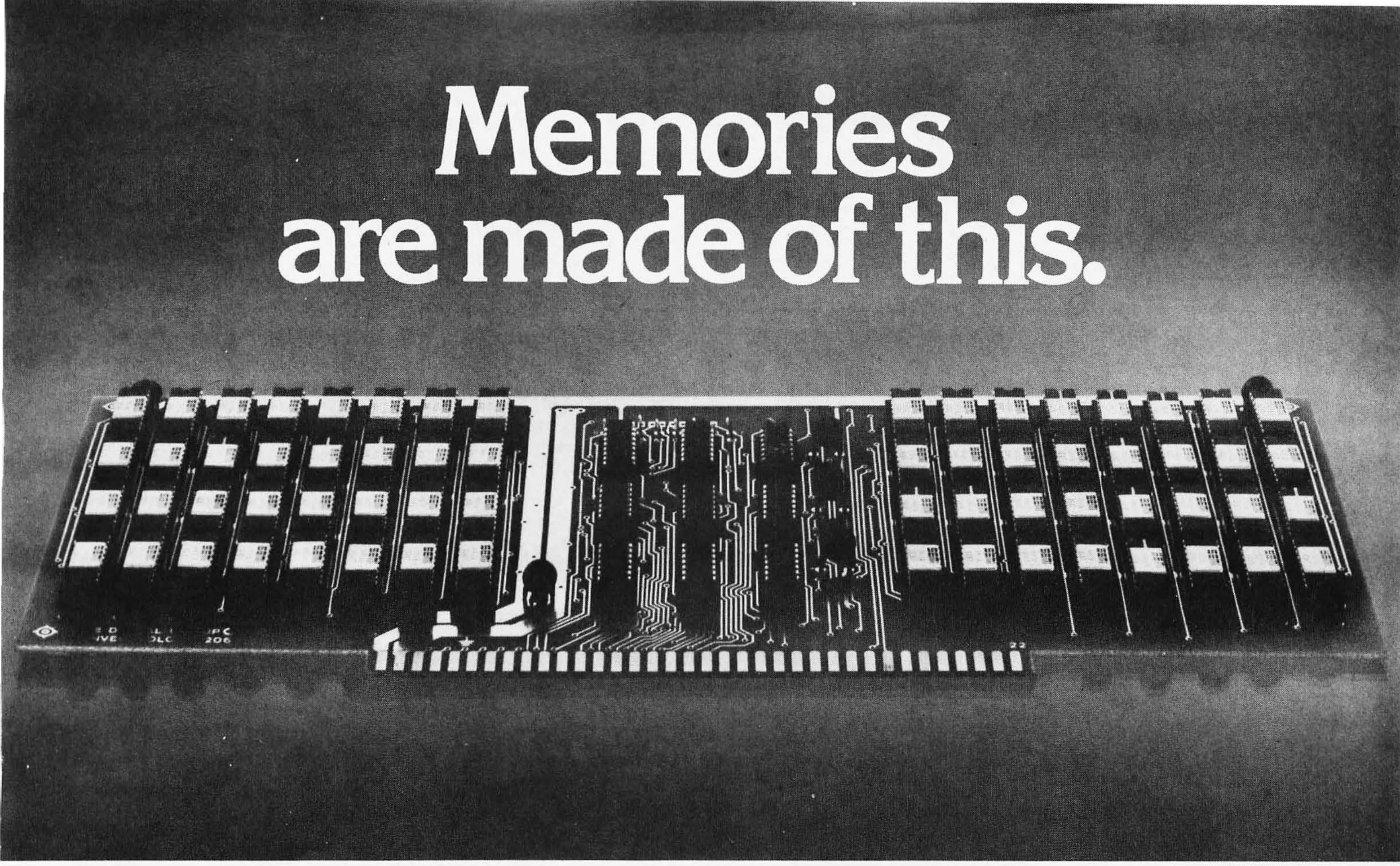
feverishly writing down all the mathematics he could. He died at 21, a monument to the stupidity of taking politics too seriously.)

A fundamental concept in all error correcting codes is "Hamming distance." Consider two code words C_1 and C_2; each code word consists of a message (data bits) together with the check bits for that message. Take the exclusive OR of C_1 and C_2; the number of one bits in the result is called the Hamming distance between C_1 and C_2. If the Hamming distance is 1, this means that C_1 and C_2 are the same, except for one bit position at which they are different. This in turn means that if it was that particular bit which was dropped, then C_1 will get mistaken for C_2, or vice versa.

A fundamental concept in all error correcting code is "Hamming distance."

On the other hand, suppose that a particular code has the property that for every pair of code words C_1 and C_2, the Hamming distance is 3 or more. Now suppose that C_1 is a code word and C_2 is not. Suppose that when a message is transmitted, then, due to some bit being dropped, it is C_2 that is received when it should have been C_1. That is, the distance between C_1 and C_2 is 1 (since only one bit was dropped). In this case the error can always be corrected. That is, of all possible code words, we can always tell that C_1 is the one we wanted. To prove this, suppose that there were another code word, C_3, that is actually the one we wanted. Then the distance between C_3 and C_2 would be 1 (since we are assuming that only one bit is dropped), and we already know that the distance between C_1 and C_2 is 1. But in this case the distance between the two code words C_1 and C_3 cannot be greater than 2, and this contradicts our assumption that two code words must have a distance between them of 3 or more.

In general, if the minimum Hamming distance between any two code words is 3, the code is a single error correcting code (although the actual correcting of the errors might, in some cases, be an elaborate and inefficient process). We can extend this immediately and say that, if the minimum distance is $d = 2e+1$, the code is an e-tuple error correcting code (usually referred to as an e-error correcting code), for any integer e. This code will not detect any more than e errors, unless we sacrifice some error correction capability. If x and y are integers with $x > y$ and $x+y+1 = d$, then we can use a code with minimum distance d, as above, to correct any y errors and simultaneously detect any x errors. As a special case of this, if $y = 0$, we can detect any d-1 errors without any error correction capability at all.■



32K. One Card. One low price. Only from the Digital Group.

Now, on only one *fully static* card, the Digital Group has squeezed in a whopping 32K of memory. Which, with a little quick addition, means a full 64K architecture now requires only 2 boards instead of 8. That's a 4-to-1 space reduction ...and leaves one extra memory slot on the Digital Group's standard motherboard still available for future products.

All this and one low price, too.
It just may be the best news of all. Our full static, assembled and tested 32K memory board is only $995. Now that's worth remembering. It's substantially less than our equivalent assembled 8K board prices. (Please note: We're initially offering this 32K board assembled only, but kit versions will soon be available, too—at even lower prices.)

Here's what you get.

Specifications:
- 32K on single card
- Speed—450ns. All of our current CPUs will operate at full-rated speed.
- Decoding—Lower or upper 32K bank
- Power— +5V only @ 4A
- Card size—12" x 5" (excluding connector fingers)

Features:
- May be intermixed on Digital Group systems with our 8K memory cards
- All data and address lines are buffered
- *Fully* static memories—EMM 4801 (450ns) or equivalent

Price:
32K board complete, assembled and tested $995.00

For all the memorable details, just fill out the coupon below. (Then all you have to remember is to mail it in.)

the digital group

P.O. Box 6528
Denver, Colorado 80206 (303) 777-7133

☐ **I promise to mail this in, so add me to your mailing list!**

Name ____________________

Address ____________________

City/State/Zip ____________________

☐ Remember me? I'm already on your mailing list, but I need the memory spec sheet desperately.

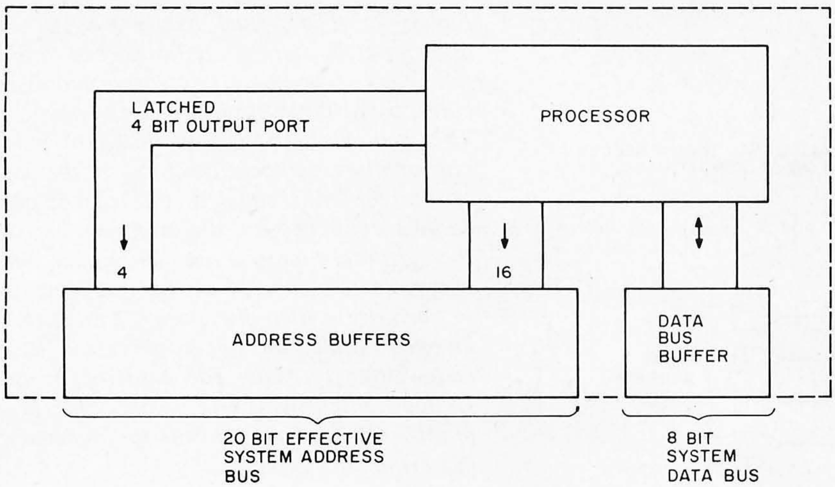
Circle 21 on inquiry card.

Figure 1: Conceptually, it's easy to expand the effective memory space of a processor by adding a latched output port which provides extra high order bits of addressing. The result is a "paged" memory technique which can address a much larger memory address space. In this example, four bits added to 16 bits yields 20 address bits, giving a microprocessor an effective addressing range of 1048576 locations. However, there are disadvantages of such a technique in this simple conception: Each 64 K region is completely isolated from its neighbors, since the IO port technique requires software switching between regions, which cannot in general be accomplished within one or a small number of instructions.

Give Your Micro a Megabyte

What is it that makes a microcomputer "micro?" There are many factors involved, but one of the most crucial is the small memory available in most hobby computers. I had scarcely gotten my Altair up and running with 8 K of memory before I wanted to write programs that exceeded its storage capabilities. Paging programs in and out from dual cassettes became a major nuisance, so I gathered my pennies and bought another 4 K memory board. In less than a month I needed even more storage. It is a fact of programming life that one must trade storage space for execution speed in programs. I tried to write a program in BASIC to play Mastermind on my Altair. Mastermind has 1296 possible moves, each move composed of four numbers. Storing this optimally would require about 5 K bytes; the BASIC code took nearly 6 K bytes, leaving me almost no space for my program. I tried to write the program to regenerate the moves each turn. This saved the space, but it took over ten minutes of computation to respond. Few human players can put up with such a wait. Don't think that this lack of memory afflicts only small machines. I work on several large computers, each of which has over 100 K bytes of storage. Programs have been written on each system which strain the memory limits. The unwritten law has it that "programs expand to fill all available space." We programming experimenters don't have much control over the structure of our processor architectures, but we do build up our own memory systems. In all systems that I have worked on, the memory was the single most expensive component. Peripherals rank second in most systems. This is the area in which we can upgrade our machines most readily from the "micro" class. This article will describe, in a general way, two methods for expanding the storage capability of a computer. These methods are widely used in big machines, but are rarely built into small machines. This need not be the case.

Present day 8 bit microprocessors typically have 16 bit address buses. This implies that a maximum of 64 K bytes can be directly addressed. Many minicomputers also have 16 bit buses, but often address up to a megabyte. How do they accomplish this apparent bit of magic? Obviously, they must come up with some more address bits somewhere. Minicomputers do it by using special registers called mapping registers. These registers contain the extra address bits. If the mapping register contents are fixed, then only 64 K can be addressed. But by dynamically changing the bits of (for example) a



Robert D Grappel
148 Wood St
Lexington MA 02173

4 bit mapping register, the processor can effectively address an entire megabyte. In minicomputers, the mapping function is usually performed automatically, when needed, by operating system software. Each task loaded in the system has access to 64 K, as determined by its mapping register. Homebrewed microprocessors can do this, too, as well as commercial systems with appropriate modifications. We need to add a register for mapping. We can't add the needed bits to the processor itself, but we can use peripheral registers as we like. Just treat the mapping register as an output device. Figure 1 shows how an 8 bit microprocessor with 16 address bits can generate an effective 20 bit address space using a 4 bit output port as a mapping register. A multitasking operating system could be written to switch up to 16 tasks, each of which has access to 64 K. With such systems, 64 K segments of data could be kept in partitions (64 K space defined by the map register), separate from the programs in which they are used. The opportunities for innovative system design are vast. As mentioned, the more memory a system has, the more complex its programs become.

"Wait a minute!", you are asking. "We were talking about microcomputers with typically 4 to 16 K. What good is extending the address space without the memory? Who could afford to build a megabyte of storage?" Good questions. The answer is the second trick used by big systems . . . it is called "virtual memory." This is quite a sleight of hand technique. Basically, it amounts to this: If you can't have memory at the address, move the address to the memory you have. Nobody expects a system to actually have a megabyte of on line storage. That is too expensive for all but the biggest machines. Disk (or tape) is also a storage medium. It is slower, but a heck of a lot cheaper for large volumes of data. Suppose that some external storage device holds a lot of data blocked into 4 K chunks. Whenever the processor asks for data, the chunk that contains that data is read into a 4 K memory from which the processor gets it. This is the essence of virtual memory: a small but fast memory which is being filled by a large but slow external store. To the program it appears that its entire address space is filled with active memory; it just has to wait sometimes. Since most programs move more or less sequentially through memory, many memory accesses are often made to the same chunk of memory. These accesses are as fast as the real memory can be. The system can have several chunks active at the same time. This, again, is the

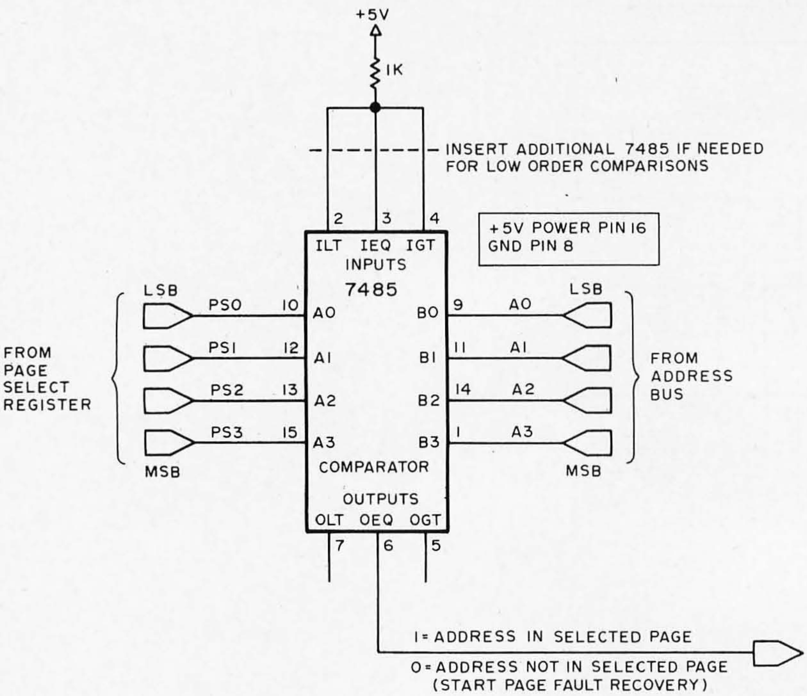


Figure 2: If a virtual memory system is implemented, there is a definite need to find out whether or not a referenced address is currently in the swapping region of memory. This can easily be accomplished by using one or more comparator chips. If the page select register is equal to the current page address reference to the swapping region, then access is normal. But if a page fault occurs (a reference to a page not presently in memory), then the paging processor is invoked while the main processor is at rest waiting for data. The 2 microprocessor strategy discussed here is but one of a number of strategies possible for implementing virtual memories.

trade-off of memory space for execution speed.

The rest of this article outlines a virtual memory system for a typical microprocessor. It will work with most popular processors. The one requirement is a READY or WAIT capability that will halt the processor at any point in an instruction upon an external signal. This is absolutely necessary to halt the processor until the desired chunk (or more properly "page") of memory has been loaded. The 8080, 6502, Z-80, TMS9900 and other processors have this feature. The 6800, unfortunately, does not.

We will assume a page size of 4 K bytes. This means that the low order 12 bits of the address bus will address the memory page. We will assume a single 4 K page of real memory for simplicity. Two processors need access to this memory. The main processor uses it, but there must also be another processor which can access the memory page to move the data into and out of it. Thus, both the address and data buses must have switches to permit either processor access to the memory. Now, we need to know if a given address lies in the current memory page. The paging processor uses an output port as a page address register. This register is compared with the upper bits of the address from the main processor. A circuit for such a page select logic is shown in figure 2. It compares the upper four bits of the address bus against the contents of a 4 bit page register. It uses 7485 4 bit magnitude comparators to test whether the address is in the page identified by the page register contents. The output goes high when the address falls outside the page. (The 7485 is cascadable, so this circuit could be extended with extra bits to generate a virtual memory space of a full megabyte. This would require a megabyte of peripheral storage, or about four floppy disks.)

Processors are cheap these days. One may as well use a micro to control the external storage. It will be idle most of the time, when the main processor is happy with the present memory page. When the page select logic indicates that a change of page is necessary, the paging processor goes into action. The main processor is made to wait. The memory buses are switched to the control of the paging processor. The paging processor reads the upper four (or more) bits of the main processor address. If the present page of memory has been written into, then the new contents of the page must be restored onto the external store. The new page must be read in from external storage and placed into the memory page. The page written latch is reset, since we will want to know if the main processor changes the contents of the page. Finally, the page address register is updated. This releases the main processor to continue processing as though nothing had happened. It never noticed that the paging processor took over. The mechanism of storage used by the paging processor is a detail of the design of the virtual memory system. It could use hard disks, floppy disks, 3M drives, digital or even audio cassettes. (A very ingenious commercial unit using a microprocessor and interpreter for the APL language runs such a virtual memory system on cassettes. Watching this little beast run its cassettes around while running a program is quite impressive. It features a 256 K byte effective address space . . . limited by the storage on a cassette. *[See the MCM/*

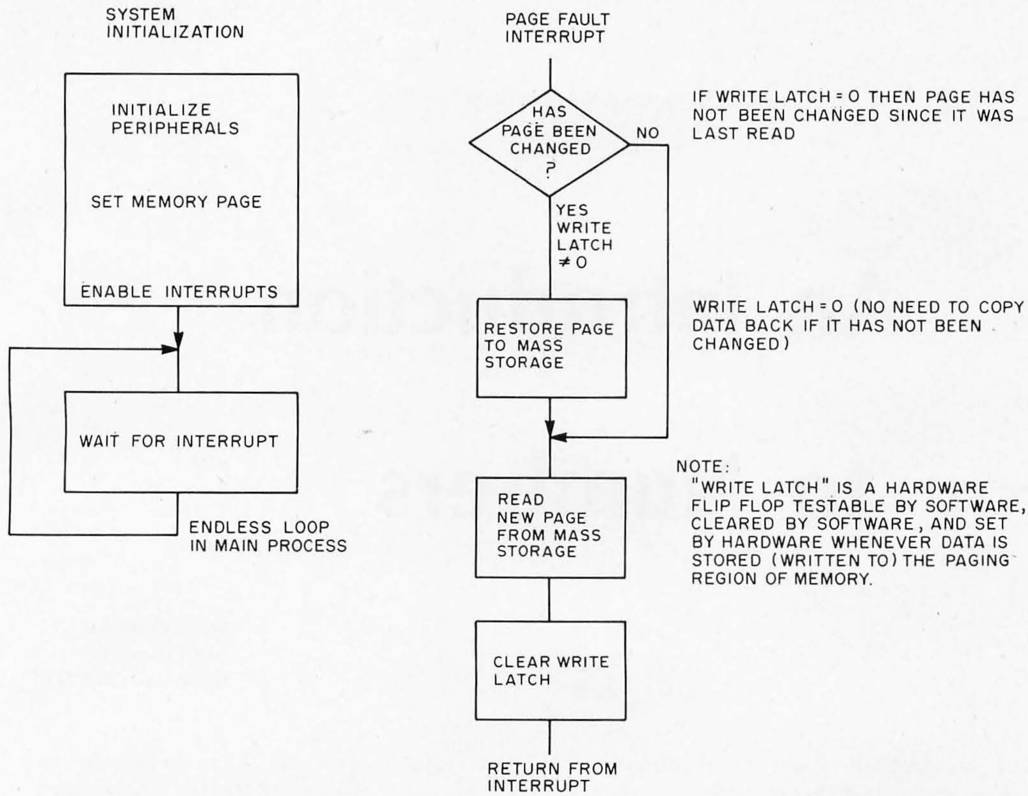


Figure 3: What to do when a page fault occurs. This is a simplified flow diagram of a virtual memory system control process, including initialization and response to a page fault interrupt. It is assumed to be running in a microprocessor dedicated to memory management.

800 sold by Microcomputer Machines Inc, 2125 Center Av, Fort Lee NJ 07024.] A simple flowchart of such a paging processor program is shown in figure 3. The main program is simply initialization followed by a wait loop. The reset function of the main processor also starts the paging processor. There is a small difficulty here: The memory page must be initialized by the paging processor before the main processor can use it. Some means must be found to keep the main processor busy while the memory page is being initialized. A oneshot triggered by the reset line (or by the paging processor), which holds the main processor in the wait state for sufficient time for the memory to be set up, will work. An output bit from the paging processor can also be used. The initial page would probably be the operating system. This brings up another interesting point. Since the entire content of memory is stored on an external, and nonvolatile medium, one can think of the entire memory space as a bit like ROM. A page can be write protected by not letting the paging processor restore that page from real memory. Such memory protection can be implemented by a ROM which is addressed by the page address lines from the main processor and which resets the page written latch. Thus, the page processor doesn't think that the page has been changed, and doesn't write on the external copy. Software in the page processor can implement program control of "write protection."

Another point is that if the main processor has ROM in its address space, one doesn't want the paging processor to waste time trying to page into that address area. A similar use of a ROM in the paging processor can inhibit paging of certain pages in which the main processor ROM is located.

With a virtual memory system, and perhaps memory mapping, there is often no need to do IO with large data blocks. Everything is, effectively, in memory address space and IO is often hidden by the paging process. This can greatly simplify programs that use lots of data.

Virtual memory systems are not especially cheap. Using a floppy disk, a system such as I describe here would probably cost around $2000 to build. A single floppy stores about 256 K bytes; so with mapping, this system would allow a quarter megabyte address space. Think of wiring together two thousand 2102s! Think of the power supply that would require! Besides, 2102 chips cost much more than a dollar each. The virtual memory space could be doubled by adding another floppy drive. It should take quite a while for you to exceed these storage limits. When you do, the system readily expands. You're not likely to byte off more than your micro can chew.■

An Introduction to Numbers

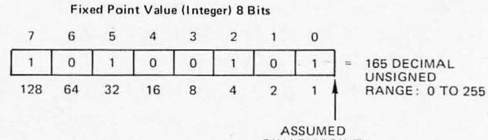
Webb Simmons
1559 Alcala Pl
San Diego CA 92111

The concept of fixed point numbers, scaled numbers and floating point numbers originated in the scientific computer environment at a time when a computer was generally considered to be either a scientific computer or a business computer rather than the general purpose computer of today. Business computers used fixed point numbers but designers felt no need to describe them as such because that was the only kind of number used. Some business computers used decimal arithmetic rather than binary arithmetic and allowed a variable amount of decimal digits for various variables and values of programs. Some business computers referred to their decimal digits as characters and regarded every character whether or not it was a decimal digit as having a decimal digit value. But here we're concerned with the wider concepts of number representation in a general purpose computer.

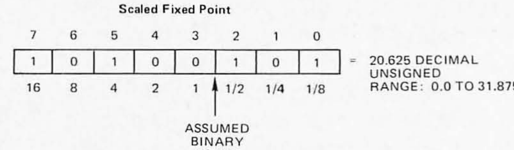
In the general purpose machine we regard all numerical values to be binary numbers in some sense. The point in fixed point and floating point is not a decimal point but is a binary point. A binary point in a binary number plays the same role as the decimal point in a decimal number. The binary number 101 (meaning 101.) has the decimal value of 5. The binary number 10.1 has the decimal value of 2.5. The binary digit to the left of the binary point has the place value, positional value, of one; the binary digit next further left has the place value of two, then four, then eight, etc. The first binary digit to the right of the binary point has the place value of one half, then one fourth, one eighth, etc. Binary 1010.1010 is decimal 10.625. Conversion is seen by adding the digit's place values 10.625 = 8 + 2 + 0.5 + 0.125.

The binary value in a register or memory location contains only binary digits as a succession of binary zeroes and ones. It has nothing in it that is explicitly a binary point. It is the responsibility of the programmer to decide the assumed position of the binary point. If the binary point is assumed to be to the right of the least significant binary digit, the value is an ordinary integer. Such a value is often called a "fixed point" number as in FORTRAN or PL/I. The binary point can be assumed to lie anywhere within the word or anywhere outside of the word. When the binary point is assumed to be fixed at any place other than at the right of the least significant bit (LSB) it is commonly called a scaled value, or scaled fixed point value. Any other fixed placement is a scaled binary number.

Fixed Point Value (Integer) 8 Bits



Scaled Fixed Point



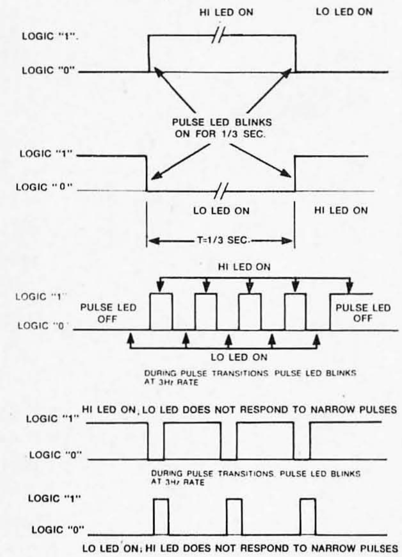
Logic Probe 1 is a compact, enormously versatile design, test and troubleshooting tool for all types of digital applications. By simply connecting the clip leads to the circuit's power supply, setting a switch to the proper logic family and touching the probe tip to the node under test, you get an instant picture of circuit conditions.

LP-1's unique circuitry—which combines the functions of level detector, pulse detector, pulse stretcher and memory—makes one-shot, low-rep-rate, narrow pulses—nearly impossible to see, even with a fast scope—easily detectable and visible. HI LED indicates logic "1", LO LED, logic "0", and all pulse transitions—positive *and* negative as narrow as 50 nanoseconds—are stretched to ⅓ second and displayed on the PULSE LED.

By setting the PULSE/MEMORY switch to MEMORY, single-shot events as well as low-rep-rate events can be stored indefinitely.

While high-frequency (5-10MHz) signals cause the "pulse" LED to blink at a 3Hz rate, there is an additional indication with unsymmetrical pulses: with duty cycles of less than 30%, the LO LED will light, while duty cycles over 70% will light the HI LED.

In all modes, high input impedance (100K) virtually eliminates loading problems, and impedance is constant for all states. LP-1 also features over-voltage and reverse-polarity protection. Housed in a rugged, high-impact plastic case with strain-relieved power cables, it's built to provide reliable day-in, day-out service for years to come.



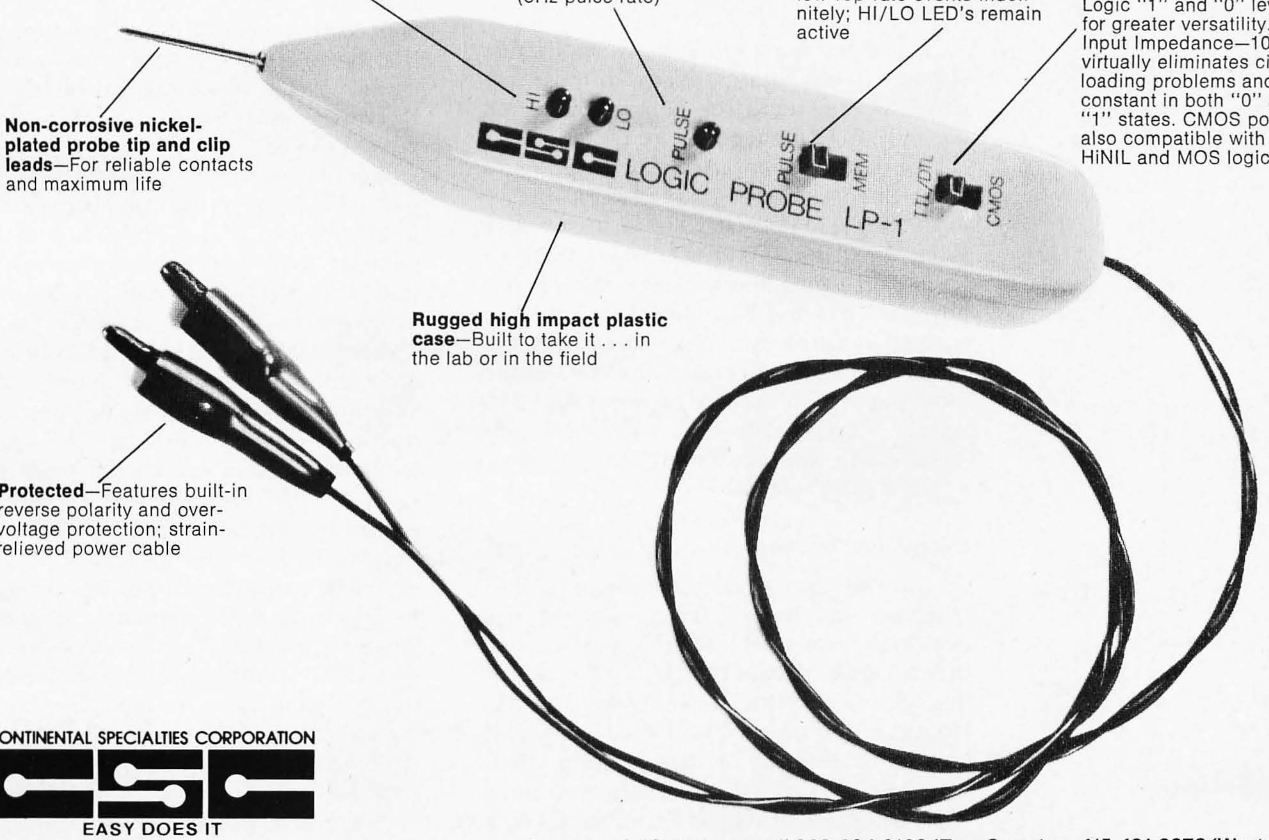
CSC'S MULTI-FAMILY LOGIC PROBE 1. AT $44.95, IT DIGS UP A LOT OF INFORMATION WITHOUT BURYING YOUR BUDGET.

PULSE LED—Lets you know what's going on—and off. Indicates positive *and* negative pulse and level transitions. LP-1 stretches pulses as narrow as 50 nanoseconds to full ⅓ sec. (3Hz pulse rate)

PULSE/MEMORY Switch—PULSE position detects and stretches pulses as narrow as 50 nanoseconds to ⅓ sec. Switch to MEMORY and it stores single shot and low-rep-rate events indefinitely; HI/LO LED's remain active

HI/LO LED's—Display level (HI-logic "1", LO-logic "0") of signal activity at node under test

Logic Family Switch—TTL/DTL or CMOS matches Logic "1" and "0" levels, for greater versatility. High Input Impedance—100K virtually eliminates circuit loading problems and is constant in both "0" and "1" states. CMOS position also compatible with HTL, HiNIL and MOS logic



Non-corrosive nickel-plated probe tip and clip leads—For reliable contacts and maximum life

Rugged high impact plastic case—Built to take it . . . in the lab or in the field

Protected—Features built-in reverse polarity and over-voltage protection; strain-relieved power cable

CONTINENTAL SPECIALTIES CORPORATION
CSC
EASY DOES IT
44 Kendall St., Box 1942 New Haven, CT 06509
TWX: 710-465-1227
West Coast office: Box 7809, San Francisco, CA 94119
TWX: 910-372-7992

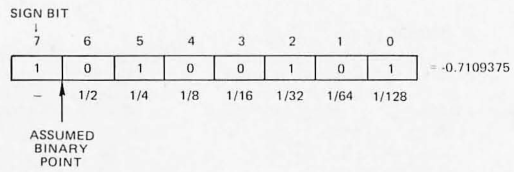
See your CSC dealer or call 203-624-3103 (East Coast) or 415-421-8872 (West Coast) 9 AM to 5 PM local time. Major credit cards accepted. Add $2.50 for shipping and handling in the U.S. and Canada on direct orders of $50.00 or less; $3.00 for orders over $50.00. On all foreign orders add 15% to cover shipping and handling.

Circle 22 on inquiry card.

The values which can be represented as an integer in one 8 bit byte are from zero to 255 when all values are considered to be unsigned, and therefore positive. In signed two's complement notation the fixed point values in one byte are the integers from −128 to +127. In either case there are 256 different values possible, of which all are integers with no fractions permitted.

Scaled fixed point binary is not often used except as a part of a floating point word. When scaled numbers were widely used in earlier computers, the binary point was frequently placed, or assumed to be placed, in the center of a long computer word of 36 bits, 48 bits or 60 bits in length. Another common placement used was at the left of the most significant bit (MSB) but to the right of the sign bit of a two's complement number. In scaled values of this type the values are always less than one and greater than minus one. Many modern large computers have instructions to facilitate operations on these fractional scaled values.

Signed Scaled Fixed Point Values



Scaled fixed point values will not figure strongly in our futures, except in those rare cases where speed or other application dependent criteria require optimization.

For signed and scaled fractional two's complement numbers, the largest binary value is 0 .1111111 and the smallest binary value would be 1 .0000000 where the digit to the left of the binary point is the sign bit; however, if we are to limit the range to fractional values, we must throw out the 1.0000000 state and treat 1.0000001 as the most negative value. This leaves us with 255 states ranging from +(127/128) to −(127/128) or performing the division, +.9921875 to −.9921875.

Extending Precision

The fact that a particular computer uses bytes for its memory storage and registers does not mean that a datum must be one byte. A unit of data can be any number of bits regardless of the computer word length. It can be 19 bits on a 13 bit word machine if you program such a construction. This would be unusual but it certainly is possible. It is customary to define the numerical data so that it will use 1,2,3 or some other whole number of bytes or words. Dividing data on memory address boundaries eases and simplifies programming.

Scaled values were useful enough for many purposes but their use was troublesome to the programmer. If a value got too large, there was danger of overflow. If a value got too small, there was a loss of significance caused by too many leading zeroes and the danger of the value becoming zero. Special scaling factors had to be used from time to time to keep the problem in hand, and of course the effects of the scaling factors had to be removed when the computations were completed. The invention of floating point numbers cured most of the scaling difficulties.

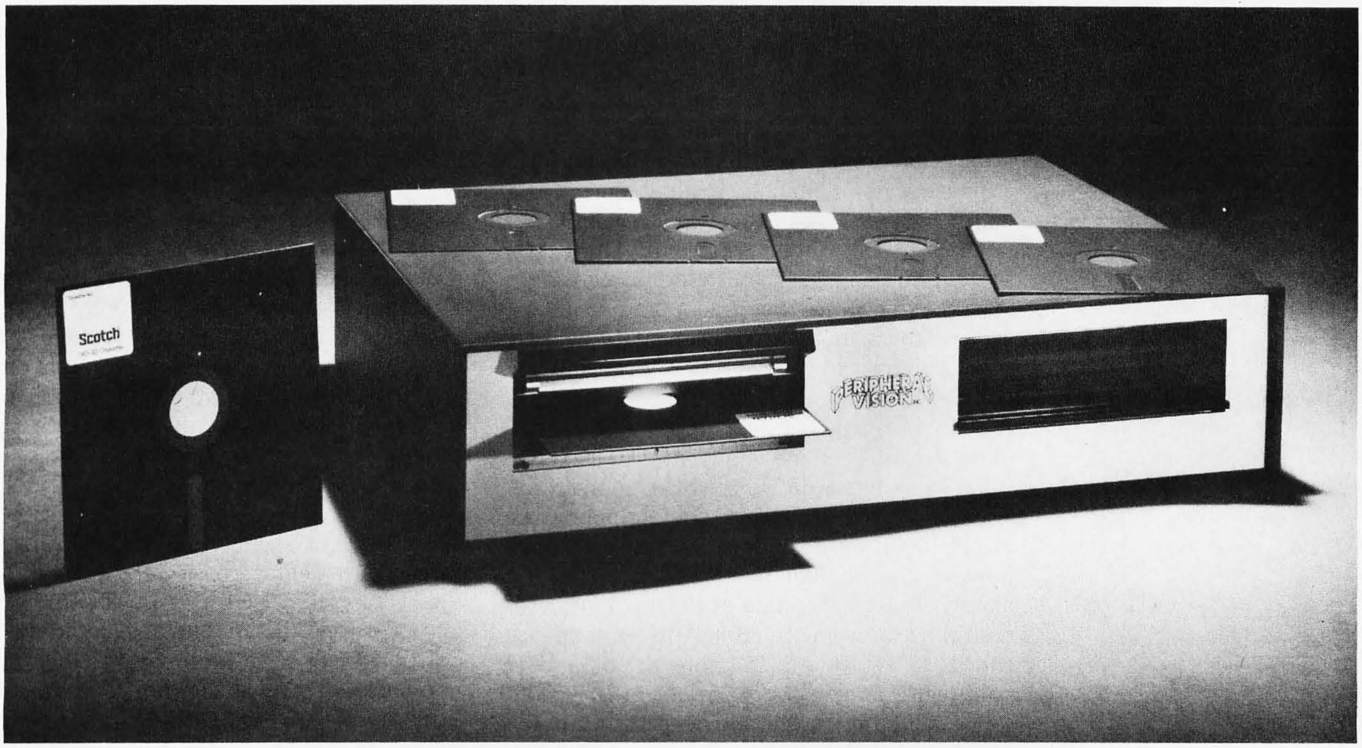
Enter Floating Points

A floating point number has two parts for each value. One part is a fraction which is a scaled fixed point number as described above. The fractional part has many names. It may be called the fraction, mantissa or coefficient. The typical floating point number has a fraction whose absolute value is always less than one. The minimum fractional value is determined by the base of the other part of the floating point number. This other part of a floating point number has been called the exponent, power or characteristic depending on whose description you read. It is the exponent or power to be applied to some base (also called the radix) that forms a scaling factor. The value of the number is the fractional part multiplied by the base raised to the power of the exponent's value.

For many years the base for the exponent part of a floating point number was almost invariably two. For a base radix of two a nominal minimum absolute value for the fractional part is one half. We set up our hardware or software to force the most significant bit of the mantissa to be one, in order to "normalize" our numbers. Similarly, for each possible base we constrain the fractional value: For a base of four the minimum fraction value is one fourth. For a radix of eight the minimum fraction is one eighth and for 16 it is one sixteenth. IBM System 360/370 uses a radix of 16 for the exponent base in floating point numbers and so do various other computers and systems. Whatever the radix for the exponent, whenever the fraction is greater than or equal to the minimum value for that base, the floating point number is said to be normalized. (In order words, it is the "normal" or "best" form.)

There is very little standardization among floating point numbers. The radix for the exponent's base may be two or it may be ten

FLIP OVER OUR FLOPPY



Only $750 from Peripheral Vision.

Peripheral Vision is a brand-new company that's dedicated to selling reasonably priced peripherals for various manufacturers' CPU's.

We think you'll flip over our first product.

It's a full-size floppy disk for the Altair-Imsai plug-in compatible S-100 BUS. And it's available for as low as $750.

Here are the features:

- 1 interface card supports 4 drives
- Stores over 300,000 bytes per floppy
- Bootstrap EPROM included—no more toggling or paper tape
- Completely S-100 plug-in compatible
- Interface cabling included
- Drive is from Innovex (the originator of the floppy concept)—assembled and tested
- Interface card design is licensed from Dr. Kenneth Welles and the Digital Group
- Disk operating system with file management system included on floppy
- Cabinet and power supply optional

Prices:	Kit	Assm.
Interface card kit and assembled and tested drive	$750	$850
Power supply— +24V at 2A	45	65
Cabinet—Optima, blue	—	85

Circle 23 on inquiry card.

Now, a little more about our company.

Peripheral Vision may be brand-new, but we have some old-fashioned ideas about how to run our business.

We know there are serious incompatibilities among the different manufacturers' peripherals and CPU's. We want to get them together. And, we want to bring significant new products to market—products consisting of everything from adaptation instructions/kits for hardware and software to major new products.

It's a tall order, but we feel we're up to the task. Peripheral Vision has already obtained a license from The Digital Group to adapt versions of some of their products to the S-100 BUS. And we're working on getting more from other companies.

Most important to our customers, Peripheral Vision is committed to helping you get along with your computer. We'll do all we can to make it easy.

Write us now for all the information on our company, our philosophy and our exciting line of products. And be prepared to flip over all of it.

PERIPHERAL VISION INC.

P.O. Box 6267 / Denver, Colorado 80206 / (303) 733-1678

Send me the works, and I just might flip over it!

Name ______________________

Address ______________________

City/State/Zip ______________________

or even 16. Within a floating point word the bits for the exponent can be either before or after the fraction. The number of bits for the exponents vary widely in different systems as do those for the fractions. There are various schemes for showing the signs for the exponents and the signs for the fractions.

The dynamic range of values allowed for a floating point number is determined primarily by the radix for the exponent and the number of bits in the exponent. The IBM System 360 and 370 hardware uses an 8 bit exponent of which one bit is used to take care of the sign for the exponent (it is not truly a sign bit but the overall effect is much the same) and one bit handles the sign for the fraction which leaves six bits to determine the value of the exponent. The largest value in six bits is 63 which, when applied to a base of 16, produces 16^{63} or approximately 7×10^{75}. The smallest positive exponent is equivalent to 16^{-64} or approximately 9×10^{-78}. These exponents must be multiplied by some fraction between 0.0625 and 1 to get the final value of a floating point word. The exponent for a Univac 1100 series computer is nine bits which loses two bits for the signs. The remaining seven bits would seem better than the remaining six for the System 370 except that the radix of the exponent is two. The largest exponent is 2^{127} or approximately 1.7×10^{38} and thus much smaller than that for the IBM version. The Univac double precision floating point word allows numbers to 2^{1024} or approximately 1.8×10^{308} which is pretty impressive.

The precision, or accuracy limit, of a floating point number is determined by the number of bits in the fraction part. Just in case you're mistakenly tempted to regard accuracy and precision as being the same, let us learn to distinguish between them. Precision relates to the ability to differentiate between value representations that are nearly the same. In terms of decimal values the precision can be to eight significant digits but if the accuracy is less than the precision then some or all of these eight digits are nonsense. Accuracy cannot exceed precision. Precision can be defined as the maximum possible relative accuracy. You cannot easily ascribe a precision to a 1 bit field because there is only one nonzero value possible.

In spite of slight conceptual error it is often convenient to regard the precision as the representional error caused by the variation of one in the least significant bit position for a field width in bits that does not include leading zero bits. Using this method for two bits we can stipulate four values, so we can say the precision is about 25%.

From this point let us decide to give precision as the precision in bits in the bit field that does not include leading zero bits. Then, for each such precision, we can compute an error that will have nothing at all to do with accuracy beyond placing a limit on the accuracy. This is equivalent to saying that a method or procedure which produces a value can be totally wrong but this does not reduce the precision in the value as it is represented. We can claim the square root of 4 to be 1.389567. This square root is quite precise but not very accurate.

When one thinks of errors it is usually errors caused by all inaccuracies rather than just those errors caused by precision of expression: However, for the remainder of this article I will take a narrow view of errors and assume they are all caused by precision only. The nominal maximum relative errors for different field widths, measured in bits, are summarized in table 1.

Our approximate rule for the maximum error in a 2 bit value gave us 25% when the true value was about 20%. As the field width in the number of bits increases the approximate rule improves and is close enough in any case. It is important to remember that the effective field width does not include leading zeroes.

The floating point fraction on the Univac 1108 (Univac calls it the mantissa) is 27 bits wide and the most significant bit is

Bits	Number of Values	Maximum Error	Decimal Digits
2	4	.250	0
3	8	.125	0
4	16	.063	1
5	32	.031	1
6	64	.016	1
7	128	8E-3	2
8	256	4E-3	2
9	512	2E-3	2
10	1k	1E-3	3
11	2k	5E-4	3
12	4k	2E-4	3
13	8k	1E-4	3
14	16k	6E-5	4
15	33k	3E-5	4
16	66k	2E-5	4
17	131k	1E-5	5
18	262k	4E-6	5
19	524k	2E-6	5
20	1M	1E-6	6
21	2M	5E-7	6
22	4M	2E-7	6
23	8M	1E-7	6
24	17M	6E-8	7
25	34M	3E-8	7
26	67M	1E-8	7
27	134M	7E-9	8
28	268M	4E-9	8
29	537M	2E-9	8
30	1G	9E-10	9
31	2G	5E-10	9
32	4G	2E-10	9

Table 1: This is a summary of the nominal maximum relative error that occurs for different field widths. The field widths are measured in bits. The maximum error column is calculated as being the maximum error or change in value that occurs if the least significant bit is lost or changed. The decimal digits column indicates the number of decimal digits which are unaffected by relative error in the representation.

always set for positive values. Thus the precision, expressed as the maximum representational error, is about one part in 10^8 and is equivalent to seven or eight significant decimal digits. The single precision floating point fraction on the IBM System 370 computers is allowed a width of 24 bits but because the exponent radix is base 16, the normalized fraction can have from none to three leading zeroes. The precision therefore varies from an error of about one part in 10^8 for 24 bits to one part in 10^7 for 21 bits. The equivalent decimal precision is about six or seven significant decimal digits. The IBM fraction is not quite as good as Univac's but the dynamic range allowed by the floating point exponent is greater. Both are greatly inferior to the CDC machines with their 12 bit exponent and 48 bit mantissa in a word of 60 bits.

It is not customary to use a signed value in the exponent part of floating point numbers. The more usual arrangement is to bias the exponent by adding a constant. The exponent range for the IBM floating word is from 16^{-64} to 16^{63}. The 16 is not shown but is assumed. IBM adds 64 to the exponents so that the floating exponent part for the value 16^{-64} is zero and for 16^{63} is 127. A floating value of one is equivalently 16X1/16=1 which yields binary 01000001000100000000000000000000 as a single, 32 bit value. In hexadecimal this is 41100000. The exponent part in hexadecimal is 41. Table 2 shows a summary of the binary and hexadecimal digit placement as used by IBM.

Negative floating values are typically formed one of two ways. The word as a whole is simply arithmetically inverted or else only the sign bit is inverted. Either of these methods is satisfactory and neither changes the dynamic range of the exponent nor the precision of the fraction.

hexadecimal	4	1	1	0	0	0	0	0
binary	0100	0001	0001	0000	0000	0000	0000	0000

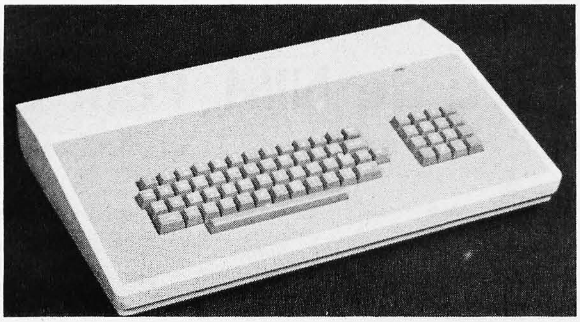
Table 2: The breakdown of a sample number of a 32 bit word into binary and hexadecimal digit groups.

Many IBM users go to double precision floating point because the precision equivalent to 6 decimal digits is not sufficient for their needs. In this case the exponent, and the dynamic range, is not changed but the fraction width is increased to 56 bits which is equivalent to about 17 decimal digits. Double precision on the Univac allows 12 bits for the exponent, versus the single precision 9 bits, and 60 bits for the fraction, versus the single precision 27 bits. Double precision on the CDC machines is almost ridiculous — the exponent is essentially repeated in the second word which allows 96 bits for the fractional part which is equivalent to more than 30 decimal digits!

If I were writing a complete programming system for an 8 bit byte machine, I would not use a 4 byte copy of the IBM floating word but would use two bytes for the exponent with an exponent base radix of two rather than 16 and four bytes for the fraction part. A 32 bit fraction is equivalent to about nine significant decimal digits and the exponent range would be ridiculously large. Or maybe one should use the IBM method but adding two bytes to the fraction part. My point here is that I personally do not like a word with only six decimal digits of precision.■

NEW

COMMERCIAL QUALITY KEYBOARD

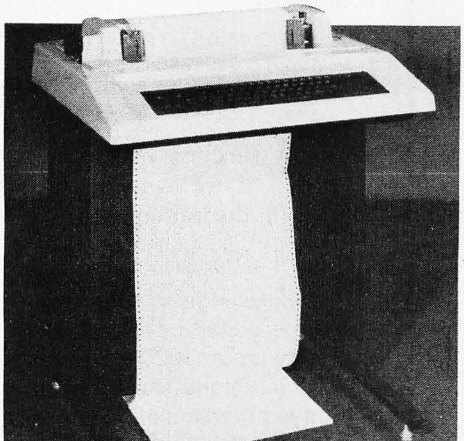


The Model SS-1 Communications Terminal is a non-contact capacitive keyboard with a guaranteed life of over 100,000,000 operations.

- **MOS/LSI Encoder with high reliability low power consumption.**
- **Key Roll Over which eliminates operator error and increases thruput.**
- **Hysteresis for tease proof operation.**
- **Solid State performance at mechanical switch prices.**
- **Tactile Feedback at the operate point.**
- **ASR-33 Array with four mode encoding.**

Keyboard Kit. **$99.95**
Enclosure (with numeric cutout) **49.95**
Numeric Pad. **34.95**
Complete Kit (with pad) **179.95**
Assembled Unit (with pad). **229.95**

DECwriter II



- **132 column printing • 30 CPS**
- **Full keyboard • Tractor Feed**

$1769.95

SOFTWARE

Attention ALTAIR DISK Owners

Now available—an Accounts Receivable System for the small user. This system is expressly designed to be run on an 8080 based microcomputer using an Altair floppy disk with Altair Basic, Video Terminal and Printer. Many features normally found only on large scale computers are included. Provided are such functions as: Adding New Customer Accounts, Deleting Dormant Accounts. Transaction Processing, and Report Generation. The Monthly Statements include both current and aged balances. Other reports generated are the Aged-Accounts Receivable and Delinquency Notices.

This easily usable system requires very little operator training. All communication is in an interactive mode with the program constantly prompting the appropriate responses. Operator errors are easily corrected and accidental data base deletions are prevented by requiring additional confirmation.

This Accounts Receivable System, although supplied ready for use, may readily be modified to include special installation-dependent functions. All major programs are written in high level language making program additions and alterations a simple task. This program module is designed to serve as the basis for a complete accounting package. Provisions are included for linking Billing Inventory Control and General Ledger modules which will be available in the near future.

We carry a full line of the following: TDL, OAE, Processor Tech., Hayden, Tarbell, IMSAI, Cromemco, Compucolor, Icom, Lear Siegler, Okidata, DEC, Javelin, Teletype, North Star, Heuristics, Peripheral Vision, TRW. Same day delivery and shipping on most items. Full, modern repair facilities on premises for complete servicing of everything we sell.

SYNCHRO-SOUND Enterprises
The Computer People
193-25 Jamaica Avenue, Jamaica, New York 11423 • 212/468-7067

Hours 9-4 daily and Saturday

Visit our new showroom . . . working units on display

Dept. B BankAmericard • Master Charge

☐ Please send me further information on: (list items)

____

______________________________ Price List ☐

Name ______________________________

Address ______________________________

City ______________ State ______________ Zip ________

Desk Top Wonders

Jeu de NIM, Peut Être?

We recently received the following from Alain Chancé in Paris. Alain is a confirmed SR-52 fan, as his letter so amply indicates. It just goes to show that enthusiasm for small systems programming is by no means limited to the US.

Alain Chancé
20 rue de Longchamp
75116 Paris FRANCE

I am an enthusiastic reader of your magazine. Since I own an SR-52 with PC-100 printer attachment, I have read "The Buried Gold in the SR-52" and "Desk Top Wonders" [*December 1976 BYTE, pages 30 and 92*] with great interest.

But I think that the register organization offers a much more fantastic ability: By the use of memory functions and program storage registers, a program can alter itself!



"That funny black caterpillar you just killed will set you back five bucks."

By Duane Bibby

To find out the correspondence between codes and program register contents, we execute the following sequence using the SR-52's 2 digit keycodes:

```
*π EE 12 STO 70 *rset *list
```

and we get:

```
display  = 3.141592654 12
printout = 000 20 001 01 002 00 003 54
           004 26 005 59 006 41 007 31
```

The following program should prove very useful to SR-52 enthusiasts:

000 0	001 0	002 0	003 0
004 0	005 0	006 0	007 0
008 0	009 *LBL	010 A	011 0
012 0	013 0	014 0	015 *rtn
016 *LBL	017 B	018 EE	019 5
020 8	021 +	022 RCL	023 7
024 1	025 =	026 STO	027 7
028 0	029 *rtn	030 0	031 0

To see how it works, perform the following sequence:

```
* π STO 05 5 STO 97
Input 0.7094336
Press B, display = 5.607094336 60
Press A, display = 3.141592654 00
```

The A function operates exactly the same way as if we had keyed in the following steps (in the learn mode):

```
*IND RCL 97
```

To replace those four steps by GTO 123 (whose codes are 41 01 02 03), we would merely input .3020141 and press B.

This unique feature opens a new dimension in programming.

For SR-52 games hunters, I offer the following version of NIM which allows up to nine rows of 1023 pawns each. (This program should be run with the PC-100 printer.)

Program Listing

SR-52 CODING FORM ~ KODEFORM ~ FEUILLE DE PROGRAMMATION

TITLE / TITEL / TITRE: JEU DE NIM PAGE / SEITE / PAGE: 2 OF / VON / DE:

PROGRAMMER / PROGRAMMIERER / PROGRAMMEUR: Alain CHANCÉ DATE / DATUM / DATE: fin août 1976

Loc. Adr. Adr.	Code Kode Code	Key Taste Touche	Comments Bemerkungen Commentaires	Loc. Adr. Adr.	Code Kode Code	Key Taste Touche	Comments Bemerkungen Commentaires	Loc. Adr. Adr.	Code Kode Code	Key Taste Touche	Comments Bemerkungen Commentaires	Labels Label Labels
000 ~~112~~	46	*LBL			00	0			00	0		A nombre de pions au début
	11	A	A		58	*dsz			42	STO		B numéro de la colonne concernée
	42	STO		040 ~~152~~	87	*1			06	6		C JEU
	06	6			56	*rtn			07	7		D non affecté
	09	9			46	*LBL		080 ~~192~~	56	*rtn		E E(x) x>0
05 ~~117~~	98	*prt			12	B	B		46	*LBL		A' 10 STO 00
	99	*pap			42	STO			19	*D'	*D'	B' $2^{(RCL00-1)}$
	36	*IND		045 ~~157~~	09	9			01	1		C' RCL00 STO67
	42	STO			08	8			00	0		D'
	09	9			98	*prt		085 ~~197~~	55	÷		E' =0 si pair, ≠0 si impair
10 ~~122~~	09	9			85	+			42	STO		Registers/Register/Memoires
	16	*A'			46	*LBL			45	y^x		00 utilisé pour l'instruction *dsz
	00	0		050 ~~162~~	16	*A'	A'		43	RCL		01 contiennent
	46	*LBL			01	1			06	6		02 le nombre
	87	*1	*1		00	0		090 ~~202~~	07	7		03 de pions
15 ~~127~~	36	*IND			42	STO			65	×		04 de chacune
	49	*PROD			00	0			36	*IND		05 des 9 rangées
	09	9		055 ~~167~~	00	0			43	RCL		06 exprimé
	08	8			95	=			00	0		07 dans le
	43	RCL			42	STO		095 ~~207~~	00	0		08 système
20 ~~132~~	06	6			09	9			95	=		09 binaire
	09	9			09	9			56	*rtn		10 0
	75	−		060 ~~172~~	56	*rtn			46	*LBL		11 contiennent
	17	*B'			46	*LBL			10	*E'	*E'	12 le nombre
	22	INV			17	*B'	B'	100 ~~212~~	85	+		13 de pions
25 ~~137~~	80	*if pos			02	2			02	2		14 de chacune
	00	0			45	y^x			95	=		15 des 9 rangées
	03	3		065 ~~177~~	43	RCL			15	E		16 exprimé
	07	7			00	0			55	÷		17 dans le
	42	STO			00	0		105 ~~217~~	02	2		18 système
30 ~~142~~	06	6			55	÷			75	−		19 décimal
	09	9			02	2			46	*LBL		Flags/Flags/Kennungen
	01	1		070 ~~182~~	95	=			15	E	E	0
	36	*IND			56	*rtn			22	INV		1
	44	SUM			46	*LBL		110 ~~222~~	52	EE		2 Utilisé
35 ~~147~~	09	9			18	*C'	*c'		53	(		3
	08	8			43	RCL		TEXAS INSTRUMENTS				4
	01	1		075 ~~187~~	00	0						

Play Instructions

1. Load in the play card both A and B sides
2. Initialize: *fix 0 *CMs *rset
3. Enter the starting position: For each non-empty row, input its number, n, $(1 \leqslant n \leqslant 9)$ and press B; then input the number of pawns, x_n, $(0 \leqslant x_n \leqslant 1023)$ and press A.
4. To play against the SR-52, input the number, n, of the row you have altered and press B; then input the number of pawns that remain in this row and press C.
5. To begin a new game go to line 2 above.

Example of Play

	1.		PRT
	4.		PRT
Starting Position	2.		PRT
	3.		PRT
	3.		PRT
	2.		PRT
Player	1.		PRT
	2.		PRT
SR-52 Replies	3.	00	PRT
	1.		PRT
Player	3.		PRT
	0.		PRT
SR-52 Replies	2.	00	PRT
	2.		PRT
Player	1.		PRT
	0.		PRT
	2.	00	PRT
	0.		PRT
SR-52 Wins			

I hope that such programs will encourage Texas Instruments to create a European SR-52 Users Club. Many people are for sure waiting for personal computer system dealers in Europe.■

A Machine Code Relocator for the 8080

When it became necessary to insert a routine between locations x and x+1 in memory, I found myself with two miserable options for manual patches.

Leor Zolman
362 Memorial Dr
Cambridge MA 02139

Many microcomputer hobbyists pass through an early stage in the development of their systems where machine language is the only mode available for programming. My system configuration at that point included an IMSAI mainframe and 4 K memory, a VDM-1 video driver and monitor, and a Tarbell cassette interface. The big problem when programming something like LIFE for the VDM in machine language occurs when it becomes necessary to insert a routine (call it routine y) between locations x and x+1 in memory. I found myself with two miserable options open:

1. Replace three or so bytes (the exact number depending on where the new instruction begins, on byte 3, 4, or 5) with a patch to routine y, placed somewhere else in memory, then restore the lost bytes at the end of routine y and jump back to location x+4, or so. This is not exactly ideal.
2. Manually relocate the program from location x+1 on down until the end of the program (unless you left plenty of NOPs to handle just such a situation — then, when you're all finished, you're left with a bunch of useless NOPs!) to make room for routine y. This involves writing the whole thing out on paper first then toggling it in. Once again, far from optimal.

My solution to this dilemma was to write a machine language program which performs the second type of relocation above quickly and painlessly when given just 11 bytes of relocation data as shown in table 1.

The program, which I call the "relocator," performs its duty in two phases. Phase 1 simply takes the block whose first address is a, and whose last address is b, then transfers it as is to the area beginning with

Label	Number of Bytes	Address	Comments
a	2	0DDD, 0DDE	First address of block to be relocated
b	2	0DDF, 0DE0	Last address of block to be relocated
c	2	0DE1, 0DE2	Destination address (ie: where the byte at location a should go)
d	2	0DE3, 0DE4	First address to have references fixed
e	2	0DE5, 0DE6	Last address to have references fixed
f	1	0DE7	Function select: 00 = fix references only 01 = move block and fix references 02 = move block only

Table 1: These six key pieces of information must be entered into the locations shown (beginning at hexadecimal 0DDD for listing 1) before relocation can be performed.

About the Author: *Leor Zolman has been programming for four years, including such applications as medical graphics for Cedars of Lebanon Hospital in his hometown of Los Angeles. He is currently working on a 2 K resident monitor for the 8080, and characterizes himself as a BASIC games enthusiast. Mr Zolman is 17 years old and a freshman at MIT.*

address c. That's it for phase 1. If the function select byte is equal to 02, then the relocator quits here and goes into an infinite loop which shows up as a known pattern in the lights of an Altair or IMSAI front panel. This is an indicator of the end of execution.

Phase 2 is the tricky part. All addresses (meaning all 2 byte operands of 3 byte instructions) which fall between items a and b above (inclusive) must have added to them the distance (number of bytes) which the block was moved. This value is easily computed by the expression c–a. For example, if a ten byte block was moved from starting location 10 to decimal starting location 75, then the value 65 must be added to any 2 byte operand in the program having a value from 10 to 20 inclusive.

The purpose of items d and e in the relocation data is to tell the relocator where it must look for such operands (references), so that they may be altered or "fixed" accordingly. Since the references may lie anywhere within the program (not necessarily just within the relocated block) d should be set equal to the very first location of the program, and e should be set to the last. The exception to this, which occurs when a data block lies in the way, will be covered later. *[To avoid the exception, the "good" programming practice of keeping data separate from executable code is highly recommended . . .CH]* Phase 2 may be executed without phase 1 if the function select is set to 0.

The beauty of this program becomes apparent when a friend having 32 K in his machine hands you a tape of a program he wrote residing somewhere up in the sky (or so it seems to you, with only 4 K) and you would like to run it. All you have to do in this case is:

1. Load in the relocator
2. Load in your friend's program in any free memory space
3. Feed the relocator this relocation data:
   a. the original first address of the program
   b. the original last address of the program
   c. the location where you loaded it (first location)
   d. same as c
   e. c + length of the program (this equals c + b – a + 1)
   f. 00 (function select to fix references only)
4. Examine the start of the relocator and hit run. When the LEDs stop flickering (about 1 second for each 1 K between

Listing 1: A relocation program for use on 8080a systems. The program takes any existing program and transfers it to any other location desired in memory with a minimum of fuss. Throughout this program, endless loops are used to signal the end of major operations. These loops will cause the data LEDs to output a constant light pattern on the front panel (line 0D37 for example). If desired, the program can be modified so that it jumps to any other desired routine at these points, or returns to your monitor. All numbers followed by an H are considered to be hexadecimal in this program (eg: 02H in line 0D35). Interestingly enough, while the program resides at starting location 0D00, it is perfectly capable of relocating itself!

Address	Hexadecimal Code	Label	Op	Operand	Commentary
0D00	31 FF 0D		LXI	SP, STACK	
03	2A DF 0D		LHLD	SBOT	get b;
06	54		MOV	D,H	move b to DE;
07	5D		MOV	E,L	
08	2A E1 0D		LHLD	DTOP	get c;
0B	44		MOV	B,H	move c to BC;
0C	4D		MOV	C,L	
0D	2A DD 0D		LHLD	SSTRT	get a;
10	C5		PUSH	B	save c;
11	CD A9 0D		CALL	COMPH	complement HL;
14	19		DAD	D	HL := b–a;
15	44		MOV	B,H	move to BC;
16	4D		MOV	C,L	
17	E1		POP	H	get c;
18	09		DAD	B	set HL equal to bottom of destination area;
19	3A E7 0D		LDA	FUNK	get f;
1C	B7		ORA	A	if f=0, then skip phase 1;
1D	CA 3A 0D		JZ	STEP 3	
20	1A	X	LDAX	D	get byte from source;
21	77		MOV	M,A	store byte at destination;
22	78		MOV	A,B	
23	B7		ORA	A	
24	C2 2C 0D		JNZ	Y	
27	79		MOV	A,C	
28	B7		ORA	A	
29	CA 32 0D		JZ	TEST	if move is over, then go to TEST;
2C	2B	Y	DCX	H	else keep a goin';
2D	1B		DCX	D	
0D2E	0B		DCX	B	
2F	C3 20 0D		JMP	X	
32	3A E7 0D	TEST	LDA	FUNK	move is now over;
35	FE 02		CPI	02H	
37	CA 37 0D	DONE	JZ	DONE	if move only, then go to DONE. (this is endless loop);
3A	E5	STEP 3	PUSH	H	save c;
3B	62		MOV	H,D	get a;
3C	6B		MOV	L,E	place a in HL;
3D	CD A9 0D		CALL	COMPH	
40	D1		POP	D	
41	19		DAD	D	DE := c–a;
42	22 E8 0D		SHLD	DISP	this is the displacement;
45	2A E3 0D		LHLD	START	get d;
48	2B		DCX	H	
49	23	LOOP	INX	H	
4A	EB		XCHG		
4B	2A E5 0D		LHLD	STOP	
4E	EB		XCHG		
4F	7B		MOV	A,E	
50	95		SUB	L	
51	7A		MOV	A,D	
52	9C		SSB	H	
53	DA 53 0D	DONE2	JC	DONE2	if reference fixing is completed, then go to DONE2. (this is endless loop);
56	06 1A		MVI	B,1AH	get 3 byte op code count;
58	11 B1 0D		LXI	D,TABLE3	address of 3 byte op table;
0D5B	1A	CHEK3	LDAX	D	
5C	BE		CMP	M	
5D	CA 7B 0D		JZ	ACT	if next byte in memory is start of a 3 byte operation, go to ACT;
60	05		DCR	B	else try next op;
61	13		INX	D	
62	C2 5B 0D		JNZ	CHEK3	if there are more 3 byte ops left to check, go back to CHEK3;

Listing 1, continued:

Address	Hexadecimal Code	Label	Op	Operand	Commentary
65	06 12		MVI	B,12H	else perform same comparison with table of 2 byte ops;
67	11 CB 0D		LXI	D,TABLE2	
6A	1A	CHEK2	LDAX	D	
6B	BE		CMP	M	
6C	CA 77 0D		JZ	SKIP	
6F	05		DCR	B	
70	13		INX	D	
71	C2 6A 0D		JNZ	CHEK2	if there are more 2 byte ops left to try, go back
74	C3 49 0D		JMP	LOOP	to CHEK2;
77	23	SKIP	INX	H	else must be 1 byte,
78	C3 49 0D		JMP	LOOP	ignore it, skip the operand of a 2 byte instruction;
7B	E5	ACT	PUSH	H	a 3 byte operation;
7C	2A DF 0D		LHLD	SBOT	get b;
7F	54		MOV	D,H	move b to DE;
80	5D		MOV	E,L	
81	2A DD 0D		LHLD	SSTRT	get a;
84	44		MOV	B,H	move a to BC;
85	4D		MOV	C,L	
86	E1		POP	H	
87	23		INX	H	HL points to B2–B3;
88	7B		MOV	A,E	
89	96		SUB	M	
0D8A	23		INX	H	
8B	7A		MOV	A,D	
8C	9E		SBB	M	
8D	DA 49 0D		JC	LOOP	if operand is > b, ignore it,
90	2B		DCX	H	
91	7E		MOV	A,M	
92	91		SUB	C	
93	23		INX	H	
94	7E		MOV	A,M	
95	98		SBB	B	
96	DA 49 0D		JC	LOOP	if operand is < a, ignore it,
99	2B		DCX	H	HL points to B2–B3 again;
9A	EB		XCHG		
9B	2A E8 0D		LHLD	DISP	get displacement value;
9E	EB		XCHG		
9F	7E		MOV	A,M	add to B2–B3;
A0	83		ADD	E	
A1	77		MOV	M,A	
A2	23		INX	H	
A3	7E		MOV	A,M	
A4	8A		ADC	D	
A5	77		MOV	M,A	
A6	C3 49 0D		JMP	LOOP	go look for more 3 byte operations;
A9	7C	COMPH	MOV	A,H	complement HL;
AA	2F		CMA		
AB	67		MOV	H,A	

Address	Hexadecimal Code	Label	Op	Operand	Commentary
0DAC	7D		MOV	A,L	
AD	2F		CMA		
AE	6F		MOV	L,A	
AF	23		INX	H	
B0	C9		RET		

Address	Hexadecimal Code	Label	Commentary
0DB1	01 11 21	TABLE3	table of 3 byte op codes;
B4	22 2A 31		
B7	32 3A C2		
BA	C3 C4 CA		
BD	CC CD D2		
C0	D4 DA DC		
C3	E2 E4 EA		
C6	EC F2 F4		
C9	FA FC		
CB	06 0E 16	TABLE2	table of 2 byte op codes;
CE	1E 26 2E		
D1	36 3E C6		
D4	CE D3 D6		
D7	DB DE E6		
DA	EE F6 FE		
DD	00 00	SSTRT	source start a;
DF	00 00	SBOT	source bottom b;
E1	00 00	DTOP	destination c;
E3	00 00	START	start phase 2 d;
E5	00 00	STOP	end phase 2 e;
E7	01	FUNK	function select f;
E8	00 00	DISP	displacement (program defined);

d and e) hit stop. You should see C3 in the data LEDs of an Altair or IMSAI.

That's it! Now the program should run right where you loaded it, unless one of the following problems occurs.

Relocating to a Lower Memory Location

The block move (phase 1) is done tail first. For a block of length n, then, byte n of the source would be transferred to byte n of the destination, then byte n–1 would be moved, then n–2, etc, up to byte 1. Therefore, relocating forward into higher memory always works, but relocating backward into lower memory fails if the difference between the source and destination addresses is not greater than the block length. In such a case, say moving a block at b to location a, an intermediate relocation must be done from b to c (where |c–a| > length of block), and then a second relocation must be performed from c to a. Note: I could have had the relocator check the direction of the move

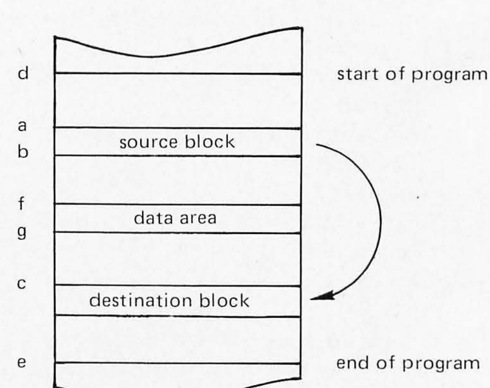


Figure 1: A special case in program relocation. The program block a thru b is to be moved to location c, but the integrity of the data block from f thru g must be preserved. The technique in this case is to change program references in two passes: up to, but not including, the data block, and then after the block. Locating all data at the end of programs will of course prevent this type of problem.

and proceed either tail or head first accordingly, but the program is long enough to cause toggling headaches as it is.

Problems with the LXI Instruction

LXI instructions in which the operand happens to equal an address in the block between a and b but doesn't actually refer to an address (LXI H, 0000 is a good example) present a problem. If you are in the process of writing the program try to use LHLD instead (or, if worse comes to worse, LHLD with MOV B, H and MOV C, L for LXI B, xxxx) and set aside two bytes of storage at the end of the program for the data.

If the program was not written by you, or if you really want to use LXI, then the only alternative is to go through your listing (if you have one!) and change the locations in question. But this problem never caused me much trouble.

Data Block Problems

Avoid relocating data whenever possible. If relocation is necessary do not try passing through it with phase 2. Here is an example of how to handle data areas: In a program residing from location d to location e, a block starting at a and ending at b is to be moved to location c. As a complication, a data block resides from location f to location g. See figure 1 for a memory map of this situation. The relocation should be performed in two runs of the relocator as follows:

Run 1. Give relocation data of a, b, c, d, f-1, 1. This moves the block and fixes all references up to location f, the start of the data area.

Run 2. Give data of a, b, c, g+1, e, 0. This simply fixes up the remaining references after skipping the data area.

*[Here again the practice of keeping data separate from executable code should alleviate this problem. . .*CH*]*

In the listing provided the relocator resides at locations 0D00 to 0DFF, but it is self-relocatable, of course. The six key items of relocation data must be set up at locations 0DDD to 0DE7 as shown in table 1, with the low byte coming first numerically in each case.

I sincerely hope that this program repays your understanding and toggling efforts with a vast reduction in relocation frustration. ■

The value of this program becomes apparent when a friend having 32 K on his machine hands you a tape of a program he wrote residing somewhere up in the sky (or so it seems to you, with only 4 K), and you would like to run it.

BASICally BASIC

(an informal introduction to BASIC)

Robert Baker
15 Windsor Dr
Atco NJ 08004

The BASIC programming language was developed in the early 1960s at Dartmouth College as a conversational, problem solving language. It has wide applications in scientific, business, and educational environments since it can be used to solve both simple and complex mathematical problems from the user's terminal. Due to the small number of clearly understandable commands and programming statements required, BASIC is one of the simplest languages to learn and use.

Like most programming languages, BASIC can be divided into two sections: elementary statements for simple programs and more advanced techniques and statements for complicated problems. As an introduction to BASIC, this article deals primarily with the elementary functions and statements recognized by BASIC as it is most frequently implemented, with program examples included to illustrate their applications. Several operating commands and a few of the more advanced features found in some BASIC systems will also be mentioned.

The specific features available on each computer system will differ slightly, forming many different dialects of the BASIC language. When writing programs to be compatible with different systems, it's general practice to use only elementary statements common to most forms of BASIC. The BASIC programming manual or equivalent documentation should always be consulted to verify the commands and statements actually available and the conventions used for a particular system.

BASIC Operating Commands

After gaining access to BASIC in whatever manner required by a particular system, BASIC will normally respond with a message indicating it is waiting for a command. The various operating commands depend on the BASIC implementation used. Several of the commands usually available are:

OLD: Load a previously saved program. BASIC may request the old program (or file) name, depending on the system.

NEW: Enter (or write) a new program from the console. BASIC may request a program (or file) name depending on the system.

LIST: Print the current program on the console. Some systems may allow printing individual lines or groups of lines.

RUN: Compile and execute the current program. Some systems may allow starting at a particular line.

SAVE, RESAVE, or REPLACE
Save the current program.

UNSAVE, PURGE, or SCRATCH
Delete the current program.

BYE, GOODBYE, or SYSTEM
Exit BASIC.

In addition, some systems may accept abbreviated commands or offer various editing commands to possibly delete lines, resequence line numbers, etc. Most BASIC interpreters will usually type READY when waiting for an operating command. New lines are inserted by simply typing a line number followed by the desired instruction terminated with a carriage return. Line numbers and individual instructions will be discussed later.

Fundamental Programming Concepts

What does it take to represent a program in a high level language such as BASIC? As

SATISFY YOUR APPETITE FOR COMPUTER KNOWLEDGE WITH SAMS COOKBOOKS

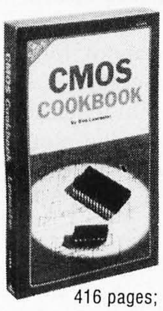
Send for the cookbooks and manuals described. Increase your knowledge of minicomputers, microprocessors, computer technology, related computer circuits and peripheral equipment. Be satisfied or your money back.



How To Buy & Use Minicomputers & Microcomputers
By *William Barden, Jr.*
This manual gives you the basics of minicomputers and microcomputers. Explains their hardware and software, the peripheral devices available and various programming languages and techniques. Allows you to decide which system is best for your needs. 240 pages; softbound. No. 21351 $9.95

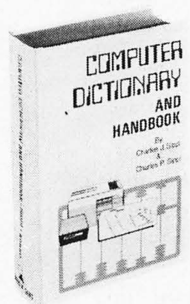


Microcomputer Primer
By *Mitchell Waite* and *Michael Pardee*
Written for the beginner in the computer field. All the basic concepts and characteristics of microcomputers are explored. The easy to understand language prepares you for further study. 224 pages; softbound. No. 21404 $7.95



CMOS Cookbook
By *Don Lancaster*
Your complete guide to the understanding and use of Complementary Metal-Oxide-Silicon integrated circuits. Gives usage rules; power supply design examples; applications; information on breadboards, testing, tools, and interface. Detailed coverage of logic and more. 416 pages; softbound. No. 21398 $9.95

The Big CMOS Wall Chart 35″ x 23″
Big, readable wall chart provides essential information on CMOS devices. No. 21399 $2.95



Computer Dictionary and and Handbook
By *Charles J. Sippi* & *Charles P. Sippi*
At your finger tips you have more than 22,000 definitions, acronyms, and abbreviations dealing with the field of data processing. Also 13 appendices cover a myriad of computer related subjects. 784 pages; hardbound. No. 20850 $19.50

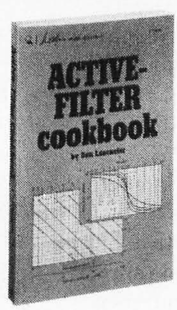


TTL Cookbook
By *Donald E. Lancaster*
You'll discover what Transistor-Transistor Logic is, how it works and how to use it. Discusses practical digital applications. You'll learn to build TTL systems that entertain, test and train. 336 pages; softbound No. 21035 $8.95

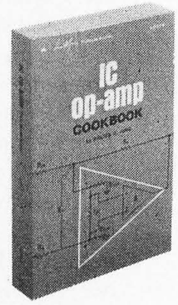
User's Guide to TTL (Wall Chart) 35″ x 23″
Shows you needed information on TTL devices at a glance. No. 20180 $2.50



TV Typewriter Cookbook
By *Don Lancaster*
Your comprehensive guide to low cost television display of alpha-numeric and graphics data for microprocessor systems, word processing, TV titling and video games. Covers configurations, memories, keyboards, techniques and much more. 256 pages; softbound. No. 21313 $9.95



Active-Filter Cookbook
By *Don Lancaster*
Dynamic coverage of active filters. What they are and how to use them. Learn to build and apply them to audio equalizers, speech therapy, psychedelic lighting and more. 240 pages; softbound. No. 21168 $14.95



IC Op-Amp Cookbook
By *Walter G. Jung*
Now one book gives you in-depth exposure to IC op amps. Covers theory and over 250 practical circuit applications. 592 pages; softbound. No. 20969 $12.95

CLIP OUT

Send your order Today!

Send books and/or wall charts checked below, $______ enclosed*. I understand that, if not completely satisfied, I may return my order within 10 days for a full refund.

☐ 21351 ☐ 21399 ☐ 21080 ☐ 20969
☐ 21404 ☐ 20850 ☐ 21313 ☐ 20715
☐ 21398 ☐ 21035 ☐ 21168

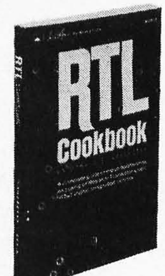
*Include sales tax where applicable. Canadian prices slightly higher.

NAME ______ please print
ADDRESS ______
CITY ______
STATE ______ ZIP ______

EJ631

Howard W. Sams & Co., Inc.
4300 West 62nd Street
Indianapolis, Indiana 46206

CLIP OUT



RTL Cookbook
By *Donald E. Lancaster*
You will learn the how and why of Resistor-Transistor Logic. Obtain useful design information and many digital applications. 240 pages; softbound. No. 20715 $5.75

Circle 26 on inquiry card.

Operator	Example	Description
+	A+B	Add A and B
+	+A	A is positive
–	A–B	Subtract B from A
–	–A	A is negative
*	A*B	Multiply A times B
/	A/B	Divide A by B
↑	A↑B	A to the power B
**	A**B	

Table 1: Arithmetic Operators. A typical BASIC interpreter incorporates an arithmetic expression parsing algorithm which recognizes the usual add, subtract, multiply, divide and exponentiate operations. Precedence is described in the text, and parentheses are used for explicit ordering of operations.

with any computer language, BASIC uses certain notations for numbers, variables and operations.

Numbers

The computer can compute the value of expressions like 12/7 or SQR(9); but while they use numbers, they are not considered numbers and may not be used in lists of constant data. Numbers are self defining values which are generally expressed in decimal form and may be positive or negative (ie: 3 or –2.79. Most BASIC languages will also allow scientific notation for numbers, where the letter "E" is used to signify 10 raised to a power. A number expressed in this form, such as aEb, would indicate "a" times 10 raised to the power "b". Thus, 0.125 may be expressed as 0.125E0 or 125E-3, and 1000 may be written as 1E3 or 10E2. With scientific notation, a number must always be present on both sides of the letter "E", and may be either positive or negative. There is usually a maximum number of digits allowed with either number notation depending on the particular computer and BASIC implementation.

Variables

Simple numeric variables are generally specified by any single letter or any letter followed by a single digit. Thus, E7 would be interpreted as a variable along with A, B, C2, and X0. Certain computer systems may allow more advanced variable labeling. But if you expect compatibility with other BASIC systems, it is best to use this restricted, simpler form.

Numeric variables represent a number with a value. The value is assigned to the variable by either LET or READ statements. An assigned value will not change until the next LET or READ statement which explicitly changes the variable. In the typical BASIC interpreter, all numeric variables are initially set equal to zero before a program is run. It's only necessary to assign an initial value to a variable when a value other than zero is desired.

Arithmetic Operators

Arithmetic computations in a language like BASIC are performed by evaluating single line formulas similar to those used in standard mathematical calculations. Any of the arithmetic operators listed in table 1 may be used to write a desired formula. In computations utilizing multiple operators, the order of precedence is determined by a very conventional set of rules:

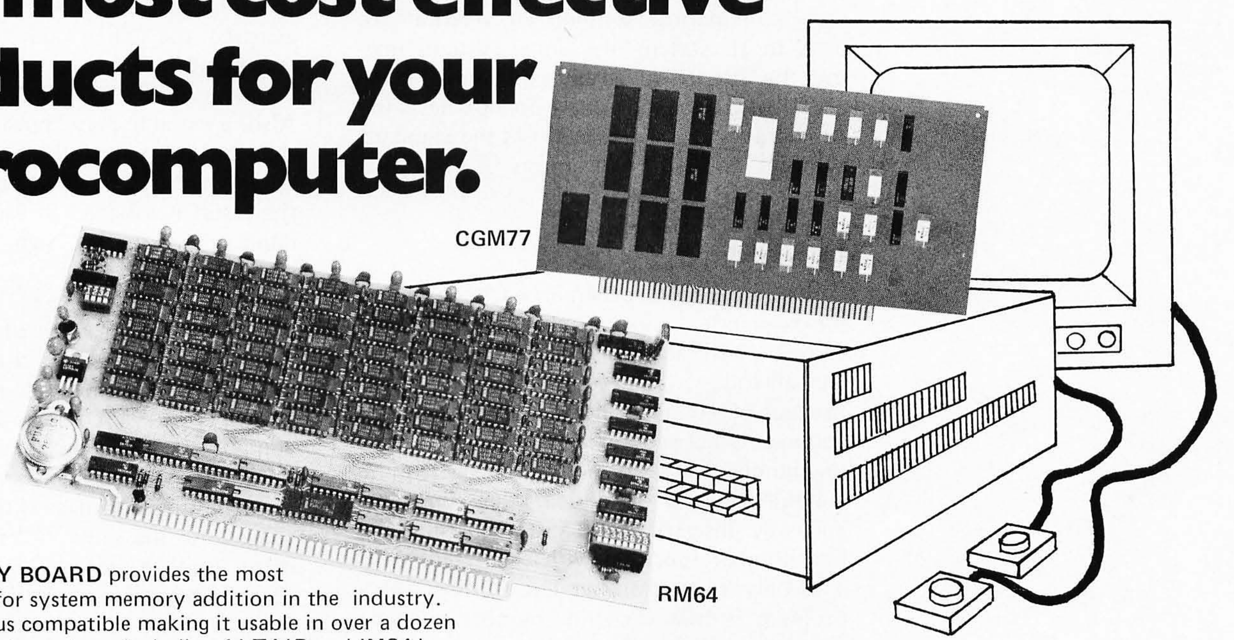
1. A formula inside parentheses is evaluated before the parenthesized quantity is used in a computation.
2. Normally, an operator cannot follow another operator. The operators + and – can, however, follow the operators *, /, ↑ or **. In this case, the + or – takes precedence over the leading operator to indicate the sign of a quantity.
3. Without parentheses in a formula, ** and ↑ take precedence over * and /, which take precedence over + and –.
4. With only * and / operators and no parentheses, BASIC performs the operations from left to right in the order they are read.
5. With only + and – operators and no parentheses, BASIC performs the operations from left to right in the order they are read.

The following examples will help illustrate these rules and how a BASIC interpreter typically executes various computations:

B+3*A**2	↔	A squared times 3, then added to B
B+(3*A)**2	↔	Product of 3 times A is squared, then added to B
(B+(3*A))**2	↔	Product of 3 times A is added to B and the sum is then squared
A/B/C	↔	Quotient of A divided by B is then divided by C
A–B–C	↔	C is subtracted from the result of B subtracted from A
A**2**3	↔	The result of A squared is then cubed

Extensys Corporation.

The most cost effective products for your microcomputer.



THE RM64 MEMORY BOARD provides the most cost effective means for system memory addition in the industry. The board is S-100 bus compatible making it usable in over a dozen different microcomputer systems including ALTAIR and IMSAI.

It comes in three configurations: 32K, 48K or 64K bytes and is completely assembled, checked out and burned in for at least 50 hours prior to shipment. This complete testing procedure enables EXTENSYS to provide you a one year warranty on parts, labor and materials (assuming no misuse of the board occurs).

On board hardware is provided for:

- **Individual memory bank address selection in 8K increments;**
- **Complete dynamic refresh logic without loss of processing efficiency while programs are running;**
- **Board select logic which allows more than one 64K byte board per system;**
- **S-100 bus compatibility including on-board voltage regulator;**
- **Memory overlap which allows memory sharing the same address space to coexist in the same system; and**
- **Write protection in 16K blocks**

THE CGM77 COMPUTER GRAPHICS MACHINE is a fully tested system that provides the basis for developing the most sophisticated video graphics imaginable. The CGM77 board is S-100 bus compatible, making it usable on over a dozen different computer systems. The only additional components needed for a complete video graphics system are 8K bytes of RAM interfaced to the microcomputer system. The EXTENSYS RM64-32K RAM board is ideally suited for this.

Special control panels are available that provide universal control for interfacing with the graphics board. Output of the board is in the form of composite sync that can be used directly by a high resolution video monitor.

The heart of the system is the EXTENSYS GRAPHICS MACHINE. This is a sophisticated piece of software developed by EXTENSYS for the rapid implementation of video graphics. The GRAPHICS MACHINE permanently resides in PROM chips that are part of the board and therefore never need to be loaded. The GRAPHICS MACHINE software actually transforms the microprocessor into a much more powerful and sophisticated computer whose instruction set is ideally suited for the development of video graphics or video games.

Contact your local computer store or order directly from EXTENSYS

B–1

extensys corporation
592 Weddell Drive
Sunnyvale, California 94086

Please place my order for the following:

QTY	DESCRIPTION	AMOUNT
____	RM64-32K byte board @ $ 895 ea.	____
____	RM64-48K byte board @ $1195 ea.	____
____	RM64-64K byte board @ $1495 ea.	____
____	CGM77 computer graphics board @ $ 375 ea.	____
	Subtotal	____
	California residents add 6½% tax	
	TOTAL	____

Shipping and handling prepaid in continental United States

Please check method of payment:
- __ Check Enclosed
- __ BankAmericard No. ________ expiration date ________
- __ Master Charge No. ________ expiration date ________

NAME ________________________

ADDRESS ________________________

CITY ________ STATE ______ ZIP ______ SIGNATURE: ____________

PHONE(INCLUDE AREA CODE) ____________

Circle 27 on inquiry card.

Arithmetic Functions

In addition to the five arithmetic operations, a fully implemented BASIC interpreter can evaluate several standard mathematical functions. The elementary functions available in a typical BASIC are listed in table 2. Any valid formula may be substituted for X in all of the functions. More advanced functions are often available and will be discussed later. Various other mathematical functions may be available depending on the individual computer system used. Sometimes extended nonarithmetic functions are also made available.

Relational Symbols

When it is necessary to compare values, six mathematical symbols of relation are used in IF statements. Some systems may use the alternate written expressions, especially when the terminals do not have the standard symbols. Table 3 lists the standard symbols and alternate forms of relational operations.

Line Numbers

In any BASIC program, each statement must include a line number as mentioned earlier. The first line is usually numbered 10 and all following line numbers are generally in steps of 10. This allows insertion of extra statements between existing lines by using a line number between the two lines where the statement is to be added. New lines are correctly inserted by BASIC by using the line number specified with each statement. The only restriction on line numbers in a program is that each line number must be greater than the preceding line number.

Elementary Programming Statements

The elementary statements of the BASIC language are identified by the first word of the statement.

Function	Description		
SIN(X)	Sine of X		
COS(X)	Cosine of X		X is usually interpreted as an angle expressed in radians on most systems.
TAN(X)	Tangent of X		
COT(X)	Cotangent of X		
ATN(X)	Arctangent of X		
EXP(X)	e raised to the power X	(e^X)	X is always interpreted as a number.
LOG(X)	Natural logarithm of X	(1n X)	
SQR(X)	Square root of X	($\sqrt{X}$)	
ABS(X)	Absolute value of X	($\|X\|$)	

Table 2: Typical BASIC Mathematical Functions. The usual implementation of a full BASIC interpreter provides this list of built-in mathematical functions. The precision and range of numbers depends upon the details of the implementation and its number representation internally.

LET Statement

The LET statement is used to assign a given number or the result of a calculation to a particular variable or group of variables. The general form of the LET statement is:

LET [variable] = [formula]*

---

*Any item shown in brackets will indicate a general form of an argument, such as:
[variable] = any variable
[line number] = any line number

With some BASIC implementations, the word LET may be optional in the statement.

Examples of assigning a value to a single variable would be:

```
10      LET X=1
20      LET A=B*2/3
```

In many BASICs, several variables may be assigned the same value by a single LET statement, such as:

```
30      LET X=Y2=B=4
```

Also, a variable may appear on both sides of the equal sign since the entire calculation is completed using "old" values of data before the result is assigned to variables on the left hand side of the "=" sign. For example, the statement:

```
100     LET X=X+1
```

will take the old value of X added to one and assign the result as the new value of X. The LET statement is not an algebraic equality, but a command to perform the computations and assign the answer to the variables specified. It is an unfortunate fact of computer language life that most languages use the same "=" symbol for assignment (data transfer) and relational tests (data comparison).

READ and DATA Statements

These statements are used to enter information into the computer for a given program. The READ statement assigns values obtained from DATA statements to the variables listed in the READ statement. A DATA statement must be present in order to use a READ statement.

Before a program is run, the typical BASIC interpreter will take all DATA statements in the order they appear and store them in a large data block. Every time a READ statement is encountered, the next available numbers from the data block are assigned to the variables in the READ statement. If the data block runs out of data, the program will stop and an error message will be printed.

READ statements are normally located near the beginning of a program since data must be read in before working with it. DATA statements may be located anywhere in a program, as long as they occur in the

Standard Symbol	Alternate Expression	Example	Description
=	=	A=B	A equals B
<	LT	A < B A LT B	A is less than B
<=	LE	A <= B A LE B	A is less than or equal to B
>	GT	A > B A GT B	A is greater than B
>=	GE	A >= B A GE B	A is greater than or equal to B
<>	NE	A <> B A NE B	A does not equal B

Table 3: Typical BASIC Relational Operators. These operators are used to specify comparisons between operands which might be numbers, variables or formulae. In certain BASIC implementations, substitutes are made when special characters are not available. For example, in a homebrew computer system using an older Baudot Teletype model, such substitutions would be a virtual requirement.

correct order. Many programmers find it convenient to place DATA statements just before the END statement at the end of the program. Each READ statement is of the form:

READ [list of variables]

While each DATA statement is of the form:

DATA [list of numbers]

Example:

```
100     READ X,Y,Z
200     DATA 1,2,3,4,5,6,7,8,9,0
```

The statements in this example would read the first three numbers of the data block (1,2, and 3) and assign them to X, Y, and Z. Always remember that only numbers, not formulas, may be used in DATA statements and 14/3 and SQR(7) are formulas.

PRINT Statement

PRINT statements are used for a variety of purposes in a typical program. While the command is called "PRINT" in many systems this means in fact "send data to the primary BASIC output device," such as a hard copy printer or electronic display device. The following example illustrates the format used to print the results of a computation or a variable value:

```
10 PRINT X, SQR(X)
```

This example would print or display the current value of X followed by the square root of X a few spaces to the right. To print a specified message, the message is enclosed in quotes and the following format is used:

```
20 PRINT "NO ANSWER"
```

This example would simply print the words NO ANSWER on a single line without the quotes. If desired, messages and values may be intermixed:

```
30 PRINT "THE SQUARE ROOT OF"
X "IS" SQR(X)
```

This statement would type the following line if the value of X were nine:

THE SQUARE ROOT OF 9 IS 3

To skip a line or issue only a carriage return and line feed, the PRINT statement is used with no arguments:

```
40 PRINT
```

With many of the more advanced BASIC systems there are various print control characters and special statements which may be used to control the spacing of the data being printed. This control may vary between systems and is beyond the scope of this article. For the present, it's sufficient to understand the primary use of the PRINT statement and to know there are often methods available to print data in any desired format.

GO TO Statement

The GO TO statement is used to unconditionally jump to some specified statement in the program other than the next sequential statement. The line being jumped to may be anywhere in the program, before or after the current line. The general form of the GO TO statement is:

GO TO [line number]

Example:

```
100 GO TO 200
```

This example would always cause the program to skip to line 200 whenever reaching line 100.

IF Statement

The IF statement is used to conditionally jump from the sequential order of statements according to the truth of some relation. The program will skip to the designated line number only if the relation specified is true. The IF statement, sometimes referred to as a conditional GO TO statement, has the general form of:

IF [formula] [relation] [formula] ,
THEN [line number]

Example:

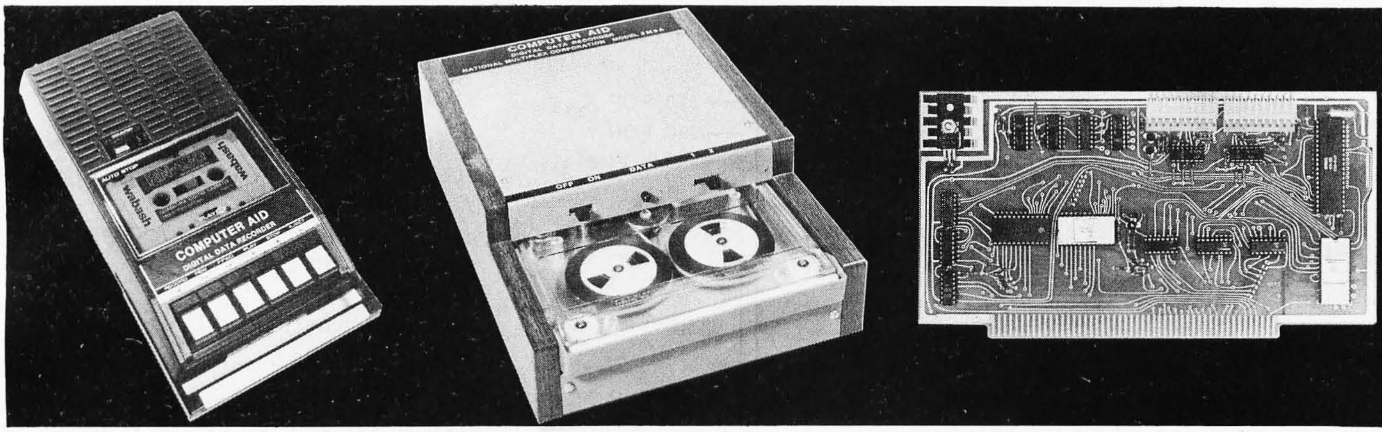
```
120 IF X=0, THEN 200
130 IF SIN(X) <=0.5, THEN 80
```

In this example, if the value of X is zero, the program will skip to line 200. If the sine of X is less than or equal to 0.5, the program will jump back to line 80. When both conditions are false, the program will go to the next line in order. The word "THEN" may be replaced with "GO TO" or the comma before "THEN" may be optional depending on the system.

ON Statement

Where the IF statement provides a two

THE LATEST IN TAPE SYSTEMS



MODEL CC-8 - $175.00 **MODEL 3M3A - $220.00** **2SIO (R) CONTROLLER $190.00 ($160.00 Kit)**

MODEL CC-8 - $175.00

4800 BAUD CASSETTE RECORDER

An ASYNCHRONOUS NRZ type Recorder with remote motor start/stop. Error rate 10^8 at 4800 BAUD. Can be used from 110 to 4800 BAUD into a UART or "Bit Banger PIA" – no clocking required. This is not an audio recorder. It takes RS232 or TTL signals from the terminal or computer and gives back the same signals. No audio interface is used. Motor start/stop is manual or through TTL or RS232 signals.

Tape speed is 3.2"/second nominal; 1.6"/sec. optional. 110 volt, 60 Hz, 5 watts. (220 Volts on special order). Can use high quality audio cassettes (Philips Type) or certified data cassettes. Can be used in remote locations from a 12 Volt battery.

Recommended for DATA LOGGING, WORD PROCESSING, COMPUTER PROGRAM RELOADING and DATA STORAGE. Especially recommended for 6800 systems, 6502 systems, 1800 systems and beginners with the 8080 systems. Manual control except for motor start/stop. 6800 or 8080 software for file or record searching available on request with order. Used by major computer manufacturers, Bell Telephone and U.S. Government for program reloading and field servicing. AVAILABILITY – Off the shelf.

MODEL 3M3A - $220.00

50 KILOBAUD CARTRIDGE RECORDER

This is a self clocking (1/1) high speed recorder. Loads BASIC in under 2.0 seconds. Recording is done at 19.2 Kilobaud. Playback at 50 Kilobaud. File or Record searching is done at 50 Kilobaud and loading is automatic. Worst case access time about 2 minutes for up to 2 megabytes on the 3M Data Cartridge.

Tape speed 10"/sec. on record, up to 30"/sec. on playback. Records one clock track and one data track on each pass (2 passes). Recording is NRZ unencoded from RS232 or TTL signals.

This recorder requires one Parallel port for motor control, and one serial port for data and clock. (Cannot be used with UART's or UART boards such as the 3P+S). Used with USART's, ACIA's or other 1/1 clocking I/O devices under software control only. No manual controls. Software for 8080 and 6800 available. Power supply is built in, 110V, 60 Hz. 220 V, 50 Hz for export.

2SIO (R) CONTROLLER $190.00 ($160.00 Kit)

PROVIDES MONITOR AND TAPE SOFTWARE in ROM TERMINAL and TAPE PORTS on SAME BOARD CONTROLS ONE or TWO TAPE UNITS (CC-8 or 3M3A)

This is a complete 8080, 8085, or Z80 system controller. It provides the terminal I/O (RS232, 20 mA, or TTL) and the data cartridge I/O, plus the motor controlling parallel I/O latches. Two kilobytes of on board ROM provide turn on and go control of your Altair or Imsai. NO MORE BOOTSTRAPPING. Loads and Dumps memory in hex on the terminal, formats tape cartridge files, has word processing and paper tape routines. Best of all, it has the search routines to locate files and records by means of six, five, and four letter strings. Just type in the file name and the recorder and software do the rest. Can be used in the BiSync (IBM), BiPhase (Phase encoded) or NRZ modes with suitable recorders and interfaces.

This is Revision 7 of this controller. This version features 2708 type EPROM's so that you can write your own software or relocate it as desired. One 2708 preprogrammed is supplied with the board. A socket is available for the second ROM allowing up to a full 2K of monitor programs.

Fits all S100 bus computers using 8080 or Z80 MPU's. Requires 2 MHz clock from bus. Cannot be use; with audio cassettes without an interface. Cassette or cartridge inputs are RS232 level.
AVAILABILITY – Off the shelf.

OVERSEAS: Export Version 220 volt 50 hz. Write factory or: Megatron-Datameg, 8011 Putzbrunn, Munchen, Germany; Nippon Automation 5-16-7 Shiba, Minato-Ku, Tokyo, Japan; Hobbydata, FACK 20012, Malmo, Sweden; G. Ashbee, 172 Ifield Road, London SW 10-9ag: Trintronics, Ltd., 186 Queen Street W., Toronto, Ontario, Canada; EBASA, Enrique Barges 17, Barcelona 14, Spain; ARIES, 7, rue Saint Phillipe du Roule, 75008 Paris; Microlem 20131, Milano, Italy; Eagle Electric, Capetown, S. Africa.

For U.P.S. delivery, add $3.00 Overseas and air shipments charges collect. N.J. Residents add 5% Sales Tax. WRITE or CALL for further information. Phone Orders on Master Charge and BankAmericard accepted.

Canadian Distributor:
Trintronics Limited
186 Queen Street West
Toronto, Canada M5V 1Z1
Tel: (416) 598-0262

NATIONAL MULTIPLEX CORPORATION
3474 Rand Avenue, South Plainfield NJ 07080 Box 288 Phone (201) 561-3600 TWX 710-997-9530

Circle 29 on inquiry card.

Legal Loops:

```
100 FOR X ...
110 FOR Y ...
.
.
.
157 NEXT Y
160 NEXT X
```

Illegal Loops:

```
NOTE CROSSOVER
780 FOR X ...
790 FOR Y ...
.
.
.
820 NEXT X
830 NEXT Y
```

Legal Loops:

```
213 FOR X ...
220 FOR Y ...
230 FOR Z ...
.
.
.
260 NEXT Z
.
.
.
300 FOR W ...
.
.
.
350 NEXT W
.
.
.
400 NEXT Y
410 NEXT Z
```

Figure 1: Nesting Diagrams: Legal and Illegal. There are legal and illegal ways to nest FOR . . . NEXT loops. The legal way to do so is to ensure that the program always has complete loops within loops. The illegal way is to have the loops cross each other. The arrows show the scope of each loop in this set of examples.

way fork in a program, the ON statement (if it is implemented) allows a multiple path switch. The general form of the ON statement is:

ON [formula],
GO TO [list of line numbers]

Any formula may be used and the instruction may contain any number of line numbers as long as the statement fits on a single line.

The value of the formula is computed and truncated to an integer. If the result is one, the program skips to the line number occurring first in the list; if the result is two, it skips to the second line number in the list, etc. If the result of the formula computation is less than one or greater than the number of line numbers listed, an error message is printed.

Example:
20 ON X–1, GO TO 100, 200, 300

This example will cause one of the following:

If X is 2, program will go to line 100
If X is 3, program will go to line 200
If X is 4, program will go to line 300
If X is less than 2 or greater than 4, an error message will be printed.

The comma preceding "GO TO" may be omitted or the word "THEN" may be substituted for "GO TO" on some systems.

END Statement

Every program must have an END statement and it must be the statement with the highest line number in the program. The general form of the END statement is simply the word "END":

9999 END

STOP Statement

The STOP statement is equivalent to a GO TO statement, where the line skipped to is the END statement of the program. The statement is simply the word "STOP":

1000 STOP

REM Statement (Remark)

The REM statement provides a method of inserting remarks and comments in a program listing. Even though whatever follows REM is ignored, the line number may be used in a GO TO, IF, ON, or GOSUB statement. The general form of the REM statement is:

REM [any desired information]
Example:
1000 REM THIS ROUTINE AVERAGES
10 NUMBERS

Elementary Programming Examples

Now that the elementary statements and functions have been introduced, three sample programs are included which will help illustrate how statements are combined to accomplish various computations or comparisons.

Loops

The simplest way to execute a sequence of instructions a given number of times is by using a program loop. The block of instructions within the loop is executed repeatedly until a given condition is satisfied. In BASIC there are two statements used to specify a loop: The FOR statement is used at the beginning of the loop and the NEXT statement at the end. The general form of the FOR statement is:

FOR [numeric variable] = [formula$_1$]
TO [formula$_2$] STEP [formula$_3$]

Depending upon the BASIC implementation, the word "BY" may be substituted for "STEP" in the FOR statement.

The accompanying NEXT statement is of the form:

NEXT [numeric variable]

The variable in the NEXT statement must be the same variable as in the FOR statement for each loop.

Loops may be nested to a level dependent on the particular BASIC implementation system, but the loops must be legally nested and not crossed. Figure 1 shows some examples of legal and illegal nesting.

The value of the numeric variable starts at the value of formula$_1$ and is changed by the

XIMEDIA PRESENTS

The SOROC IQ 120

CURSOR CONTROL. Forespace, backspace, up, down, new line, return, home, tab, PLUS ABSOLUTE CURSOR ADDRESSING.

TRANSMISSION MODES. Conversation (half and full Duplex) PLUS BLOCK MODE – transmit a page at a time.

FIELD PROTECTION. Any part of the display can be "protected" to prevent overtyping. Protected fields are displayed at reduced intensity.

EDITING. Clear screen, typeover, absolute cursor addressing, erase to end of page, erase to end of line, erase to end of field.

DISPLAY FORMAT. 24 lines by 80 characters (1,920 characters).

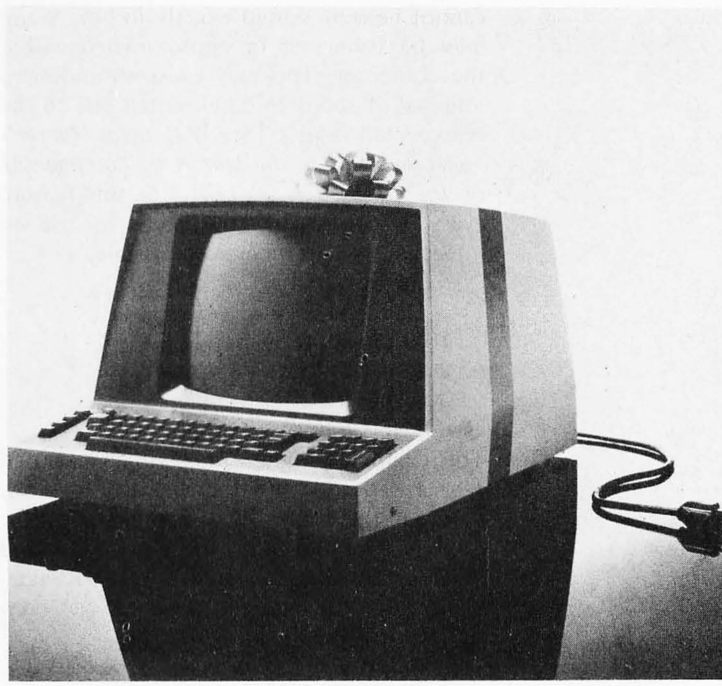
CHARACTER SET. 96 characters total. Upper and lower case ASCII.

KEYBOARD. 73 keys including numeric key pad.

REPEAT KEY. 15 cps repeat action.

DATA RATES. Thumbwheel selectable from 75 to 19,200 baud.

SCREEN. 12 inch rectangular CRT – P4 phosphor.



SPECIAL INTRODUCTORY PRICING

Kit $ 995.00 **Assembled $ 1,295.00**

(Price includes block mode, lower case and 24 line options.)

Specials of the Month

North Star **Micro Disk** with power supply and cabinet . . . . . Kit – $699
Assembled – $799

IMSAI I-8080 with **TDL** ZPU . . . . . . Kit – $825
Assembled – $975

Digital Systems FDS Disk Drive with **CP/M** Software (assembled only) . Single – $1,750
Dual – $2,350

Mountain Hardware **PROROM** . . . . . Kit – $145
Assembled – $210

Vector Graphic 8K RAM . . . . Kit – $235
Assembled – $285

NOW WE'RE TOLL FREE

800-227-4440

(in California, Hawaii, and Alaska, call collect: 415/566-7472)

XIMEDIA OFFERS A FULL RANGE OF PRODUCTS FOR THE PERSONAL COMPUTER ENTHUSIAST AND THE SMALL SYSTEM DESIGNER. LET US QUOTE ON ALL YOUR HARDWARE AND SOFTWARE NEEDS.

OUR RETAIL STORE – THE COMPUTERIST™ – IS NOW OPEN IN SAN FRANCISCO. CALL US FOR DIRECTIONS.

XIMEDIA

1290 24th Avenue, San Francisco, CA 94122
(415) 566-7472

Circle 30 on inquiry card.

COD orders freight collect. Orders with payment shipped prepaid. California residents add 6% sales tax. Please allow 3 weeks for delivery.

value of formula3 (either positive or negative) each time BASIC completes an execution of the loop. If the word "STEP" and formula3 are omitted, the step value defaults to positive one. It should be pointed out, however, that the control variable may be changed within the body of a program loop if desired. Exercise care when using noninteger step values since noninteger numbers cannot be represented exactly in binary and may be truncated or approximated within the computer. This may cause an undesired number of loops to be executed due to the unexpected results. [*See W Douglas Maurer's "Software Bug of the Month #2" on page 81 of the July 1976 BYTE.*] For this reason, every effort should be made to use an integer step value whenever possible.

Example:

```
5 LET F=1
10 FOR X=1 TO 10 STEP 1
20 LET F=F*X
30 PRINT X,F
40 NEXT X
```

This example program loop will be executed 10 times as X is incremented by one between the values of one and 10. During each pass of the loop, the numbers X and X-factorial will be printed. The FOR statement in this example could have been shortened to:

```
10 FOR X=1 TO 10
```

since the step value was one.

Subroutines

When a particular sequence of statements is to be used repeatedly in several different places within a program, they may be written as a subroutine. A subroutine is a completely self-contained program accessed by the main program or other subroutines using a subroutine calling convention.

In BASIC, subroutines are entered (or called) by using GOSUB statements anywhere in the main program (or other subroutines) of the general form:

GOSUB line number

The GOSUB statement is similar to a GO TO statement except the computer saves the location of the GOSUB statement before going to the subroutine. Statements are then executed sequentially until a RETURN statement is encountered. The RETURN statement directs the computer to return to the line following the GOSUB statement that called the subroutine. It consists simply of a line number and the word "RETURN":

```
1000 RETURN
```

and is usually the last line of a subroutine.

Subroutines may appear anywhere in the main program but should only be entered with a GOSUB statement and exited via a RETURN statement. A subroutine may actually contain several RETURN statements as long as one will be used. By using a GOSUB statement within one subroutine to call a second subroutine, subroutines may be nested. The level of subroutine nesting is sometimes limited depending on the particular BASIC implementation used.

Example:

```
10 LET L=4
20 PRINT "THIS PROGRAM PRINTS"
30 GOSUB 190
40 PRINT "THIS MESSAGE WITH"
50 GOSUB 190
60 PRINT "FOUR BLANK LINES"
70 GOSUB 190
80 PRINT "BETWEEN PRINTED
     LINES."
90 GOSUB 190
100 REM Then the numbers 1 to 10 are
     printed with the
110 REM number indicating the number
     of blank lines
120 REM following that line.
130 FOR L=1 TO 10
140 PRINT L
150 GOSUB 190
160 NEXT L
170 PRINT "DONE"
180 STOP
190 REM Subroutine to print "L"
     blank lines.
200 IF L >0, THEN 220
210 RETURN
220 FOR X=1 TO L
230 PRINT
240 NEXT X
250 RETURN
260 END
```

This example illustrates how subroutines are utilized and the use of multiple RETURN

Sample Program #1:

This program reads a group of numbers from a data block and prints the average of the numbers read. The number 9999 is used as a dummy item in the data block to indicate the end of the data block. The individual numbers of the data block are read until 9999 is found. Since all variables are initially set equal to zero until assigned another value, there is no need to include a statement to initialize S and C to zero (LET S=C=0).

Program	Explanation
10 REM print average of numbers	Remark
20 Read X	Read a number X
30 If X=9999, then 70	Check for end of data if X=9999
40 Let S=S+X	Add X to sum
50 Let C=C+1	Increment the data count C
60 Go to 20	Go back for more data
70 Print S/C "is the average of" C "numbers"	Compute and print average
80 Data 5,7,3,9,27,54,31,9999	Data
90 End	End of program

statements within a single subroutine. It also shows how the line number of a REM statement may be used in a GO TO, IF, ON, or GOSUB statement as mentioned earlier.

Every time a "GOSUB 190" statement is encountered in this example, the program will go to the subroutine starting at line 190. If the value of L is less than or equal to zero, the computer will return to the main program. Otherwise, the subroutine will print a number of blank lines equal to the value of L before returning to the main program.

Lists and Tables

In addition to the limited number of simple variables, BASIC usually allows the capability to designate the elements of lists or tables. Simple variables are called unsubscripted variables while list or table elements are called subscripted variables. Subscripted variables may be used anywhere an unsubscripted variable would normally be used, except immediately after "FOR" in a FOR statement or in a NEXT statement. Several special matrix instructions and functions will be discussed later.

The name of each list or table must be designated by a single letter. The individual elements of a list (also called a vector) are specified by the list name followed by a subscript in parentheses: A(0), A(1), . . . A(n) while table (also called a matrix) elements are specified by double subscripts: B(1,1), B(1,2), . . . B(m,n). The subscript form is flexible and may be either a constant or any legal expression as long as the subscript value is not less than zero. The single letter denoting a list or table name may also be used as a simple variable without confusion. The same letter may not, however, be used to denote both a list and a table in the same program since a list is actually a single column table.

Sample Program #2:

This program will determine the smallest and largest numbers of a given data block and print the values. The first entry in the data block indicates the number of entries to read from the data block. (An alternative would be to use end of data indicator.)

Program	Explanation
10 Let S=10E6	Set smallest number to something big
20 Read C	Get number of entries
30 Read N	Read data (once for each number)
40 If N >=S, then 70	Jump if larger than old smallest number
50 Let S=N	Set new smallest number equal to this number
60 Go to 90	Continue (skip largest test)
70 If N<=L, then 90	Jump if smaller than old largest number
80 Let L=N	Set new largest number equal to this number
90 Let C=C-1	Decrement data count
100 If C <> 0, then 30	If C not equal zero then continue
110 Print "largest number is" L	Print answers
120 Print	With blank line
130 Print "smallest number is" S	Between them
140 Data 8	Number of data entries
150 Data 2,7,42,74,21,61,47,29	Data
160 End	Program end

Example:

```
10 FOR I=0 TO 2
20 FOR J=0 TO 3
30 READ M(I,J)
40 NEXT J
50 NEXT I
60 DATA 1,2,3,4
70 DATA 5,6,7,8
80 DATA 9,10,11,12
```

This example routine shows how a 3 by 4 table may be entered into a program using elementary statements and how the table entries are specified.

Storage space is automatically reserved for any list or table with subscripts (typically) of 10 or less (the exact number depends on the implementation). For larger subscripts, space must be reserved by using a DIM statement. A DIM statement may appear on any line before the END statement, but it is normally placed at the beginning of the program. A DIM statement may also be used to reserve *less* space for a small list or table when a large program requires more program space.

The general forms of DIM statements for lists or tables are:

DIM [list name] (maximum subscript)
DIM [table name] (maximum row subscript, maximum column subscript)

Separate DIM statements are not needed for each list or table; several lists or tables may be specified in a single DIM statement by separating them with commas.

Example:

```
10 DIM V(15)
20 DIM M(20,20)
```

May also be written in a single DIM statement as:

```
10 DIM V(15),M(20,20)
```

This example reserves space for 16 items in vector V:

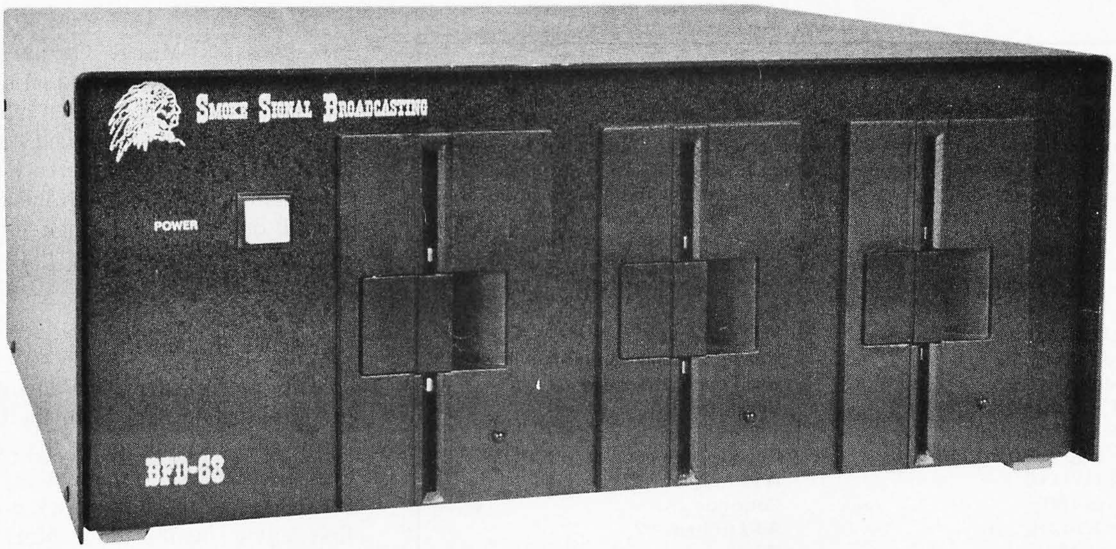
V(0), V(1), V(2), . . . V(15)

and 441 items in matrix M:

20 + 1 rows and 20 + 1 columns

Advanced Statements and Functions

This section will briefly introduce the more advanced and specialized features of BASIC. These features may or may not be available in a particular BASIC implementation, whereas the elementary statements and functions already presented should always be found. Refer to the programming manual for your particular BASIC inter-



- ANY NUMBER OF FILES MAY BE OPEN (IN USE) AT ONE TIME
- THE NUMBER OF FILES AND SIZE OF FILES IS LIMITED ONLY BY THE SIZE OF THE DISC
- MERGING FILES REQUIRES NO EXTRA DISC SPACE
- NO WAITING FOR THE DISC TO RE-PACK
- LONGER DISC LIFE — MORE EVEN DISC WEAR

Our Basic Floppy Disc System (BFD-68) must, in all modesty be called superb. It comes completely assembled with a disc controller that is plug compatible with the SWTPC 6800. In fact all our products use the 6800 standard SS-50 (Smoke Signal 50) bus used by SWTPC. The cabinet and power supply are capable of handling up to 3 Shugart Mini-Floppy Drives. One drive is included in the price of the BFD-68 and others may be added easily at any time. Or you may save money by ordering the dual-drive BFD-68-2 or triple drive BFD-68-3 (pictured). Price: BFD-68 $795, BFD-68-2 $1169, BFD-68-3 $1539, SA-400 Drive $390.

The BFD-68 includes our Disc Operating System Software. Our software provides for a soft-sectored disc format consisting of 128 bytes per sector, 18 sectors per track and 35 tracks per disc. The software provides direct commands to name and rename files, transfer memory to disc and disc to memory and to automatically jump to the starting location of any program loaded from disc to memory. The direct command names are: RUN, GET, GETHEX, CLOSE, SAVE, DELETE, APPEND, RENAME, COPY, LIST, FIND, LINK and PRINT. In addition, the Disc File Management subroutines are available to create files under your program control.

A bootstrap PROM is included on the controller board to initiate the Disc Operating System which loads into a 4K memory board located at 7000 or optionally at D000. Thus, you can be up and running from a cold start in just a few seconds.

SUPER SOFTWARE

Free patches are provided for SWTPC BASIC version 2.0 and Co-Resident Editor/Assembler. These patches allow the SAVE and LOAD commands to work with the disc or the cassette at your option.

SUPER EDITOR: Smoke Signal Broadcasting now has its own editor. It is a content oriented editor with string search and block move capability. Changes may be made by referring either to line number or string content or a combination of references. Naturally, it is designed for file transfers to and from the BFD-68. Price: SE-1 $29 on diskette.

ALL OUR PRODUCTS EXCEPT THE PS-1 ARE COMPLETELY ASSEMBLED.

SUPER ASSEMBLER: Inputs source code from file on the BFD-68 disc system and outputs object code to disc file. Assembly listings include alphabetized and tabulated symbol table. Price: SA-1 $29 on diskette.

Complete source listing included for both editor and assembler. Order both for $53 and save $5.

SMARTBUG — A CURE FOR MIKBUGITIS: A super smart Motorola-Mikbug replacement that preserves almost all Mikbug entry locations so your present programs will run without modification. Uses ACIA instead of PIA and includes many additional features including a software single-step trace command. A SMARTBUG listing is included and object code is provided on a 2708 free with each P-38 series board purchased. Source listing available separately for $19.50.

NEED A FULL SIZE FLOPPY? Our P-38-FF is a plug-in interface card to the ICOM Frugal Floppy™. It includes all the features of the P-38-I plus one 2708 EPROM containing the ICOM bootstrap software. Just plug the P-38-FF into your SWTPC 6800 and your ICOM into the P-38-FF and you're ready to use the Frugal Floppy and ICOM's 6800 software package. Price $299.

Our P-38 is an 8K EPROM board containing room for 8 2708's. Or, you may use it to hold up to 7 2708's plus your Motorola Mikbug or Minibug II ROM. The P-38 addressing is switch selectable to any 8K location. Price $179.

The P-38-I contains all the features of the P-38 plus an interface to the Oliver Paper Tape Reader and our EPROM Programmer. Price $229.

The PS-1 Power Supply Kit provides plus and minus 16 volts required for the P-38 series boards. Also, it allows a wiring modification to be made to the 8 volt supply that will increase its output by one volt. Price $24.95.

Our M-16-A is a 16K single power supply STATIC RAM memory system. The M-16-A is fully buffered and requires only half the power of a similar size system using low-power 2102's. With the M-16-A, you can expand your system to 48K and still have room left for one of our EPROM boards. The M-16-A is switch selectable to any 4K starting address and hardware write protect is included. Quick delivery. Price $595.

BANKAMERICARD, VISA AND MASTER CHARGE WELCOME.

SMOKE SIGNAL BROADCASTING

P.O. Box 2017, Hollywood, CA 90028 • (213) 462-5652

Sample Program #3:

This simple inventory program for three products illustrates the use of the ON statement by reading an item code number followed by an item count from the data block. The ON statement decodes the item code to determine what total to add the new count to. When done, the program prints all the current inventory totals.

Program	Explanation
10 REM inventory	Program name
20 Read N,I1,I2,I3	Read data count and original totals
30 Read I,C	Read code and count
40 If I> 3, then 130	Check for valid code
50 If I < 1, then 130	
60 On I, go to 70,90,110	Decode item code
70 Let I1=I1+C	Add to item #1
80 Go to 150	Continue
90 Let I2=I2+C	Add to item #2
100 Go to 150	Continue
110 Let I3=I3+C	Add to item #3
120 Go to 150	Continue
130 Print "invalid item code"	Print error message
140 Stop	Exit program
150 Let N=N-1	Decrement data count
160 If N <> 0, then 30	Finish data
170 Print "item #1 =" I1	Print totals
180 Print "item #2 =" I2	
190 Print "item #3 =" I3	
200 Print	Blank line
210 Print "total items =" I1+I2+I3	Combined total
220 Data 6,1020,1349,714	Original totals
230 Data 1,12,1,7,3,27,2,11,3,212,1, 136	Data
240 End	Program end

Notice how this program checks for an illegal value before the ON statement. An invalid item code causes a message to be printed and the STOP statement terminates the program.

preter to be sure the desired functions and/or statements are available.

INPUT Statement

The INPUT statement acts like a READ statement except data is entered from the console keyboard rather than from DATA statements. Whenever BASIC encounters an INPUT statement it types a question mark (?) to indicate it is waiting for operator input. The user then types the desired input as it would normally appear in a DATA statement terminated by a carriage return. INPUT statements are usually combined with PRINT statements to indicate what value is desired:

```
10 PRINT "NUMBER OF DAYS";
20 INPUT D
```

The semicolon at the end of the PRINT statement causes the question mark to be typed on the same line as the message. Normally, the question mark would be printed on a separate line. In this example, the words NUMBER OF DAYS without quotes would be printed followed by a question mark. The number typed followed by a carriage return would be assigned to the variable D. The data entered via an INPUT statement is not saved with the program so it should only be used when entering small amounts of data or data that is unknown until the program is run.

RESTORE Statement

The RESTORE statement typically permits reading data from the DATA statements of a program more than once. Whenever a RESTORE statement is encountered, BASIC resets the data block pointer to the first entry in the data block. The next READ statement then starts reading the data block all over again.

SIGN Function

The sign function:

SGN(X)

returns a value indicating the sign of the argument specified. The value one is assigned for any positive number, zero for zero, and −1 for any negative number.

Example:

SGN(2.75) = 1
SGN(0) = 0
SGN(−0.25) = −1

The sign function can be combined with an ON statement to give a three way branch depending on the sign of a number:

```
60 ON SGN(X)+2,
GO TO 100, 200, 300
```

This instruction will jump to 100 if X is negative, 200 if X equals zero, or to 300 if X is positive.

INTEGER Function

The integer function:

INT(X)

returns the greatest integer of X that is less than or equal to X.

Example:

INT(2.98) = 2
INT(−2.05) = −3
INT(4) = 4

The integer function may be used to round numbers to the nearest integer by adding 0.5 to the number:

INT(X+0.5)

It may also be used to truncate a number to any specific number of decimal places (n):

INT(X*10**n+0.5)/(10**n)

Random Numbers

Most forms of BASIC provide a means of generating a random number between zero and one. This function is generally used to simulate events that happen in a somewhat

CP/M

LOW-COST MICROCOMPUTER SOFTWARE

CP/M is a low-cost control program for microcomputers which brings together recent advances in computer and peripheral technology. CP/M is an advanced disk operating system designed for use with IBM-compatible diskette-based computer systems which employ the Intel 8080 microcomputer. Previously available only to OEM's, CP/M has been in existence for over three years in various manufacturers' products, and has undergone extensive field testing. The functions of this software package include named dynamic files, program editing, assembly, debugging, batch processing, and instantaneous program loading, resulting in facilities similar to popular timesharing services. CP/M is an "unbundled" software package which can be easily adapted to any 8080 or Z-80 computer system with at least 16K of main memory and one or two IBM-compatible disk drives. The field-modification manual provided with CP/M has been used successfully by many CP/M users in customizing CP/M for their own hardware configurations.

DIGITAL RESEARCH

Please send me the following:

- ☐ **CP/M Documentation** (Complete set of 6 manuals) for **$25.00.**
- ☐ **CP/M Documentation** and **CP/M System Diskette** for **$70.00.**
- ☐ Free **CP/M brochure** and **price list.**
- ☐ **Information** concerning high level languages and optional packages.

NOTE: IMS Associates and Digital Systems distribute CP/M with their hardware systems.

☐ Bank Americard No. ______________ Exp. date ________ ☐ Master Charge No. ______________

Exp. date ________ ☐ Check or M.O. enclosed.

California residents add 6% sales tax. Amount enclosed ______________

Name ______________ **Address** ______________

City ______________ **State** ______________ **Zip** ________

DIGITAL RESEARCH Post Office Box 579 • Pacific Grove, California 93950

Circle 111 on inquiry card.

MAT READ a,b,c . . .	Read the specified matrices or vectors, dimensions previously specified.
MAT M = ZER	Set all components of matrix M equal to zero.
MAT M = CON	Set all components of matrix M equal to one.
MAT M = IDN	Set up square matrix M as an identity matrix.
MAT PRINT a,b,c . . .	Print the specified matrices or vectors.
MAT INPUT V	Call for the input of a vector.
MAT B = A	Set matrix B equal to matrix A.
MAT C = A+B	Set matrix C equal to the sum of matrices A and B.
MAT C = A–B	Set matrix C equal to the result of matrix B subtracted from matrix A.
MAT C = A*B	Set matrix C equal the product of matrix A multiplied by matrix B.
MAT C = TRN(A)	Set matrix C equal to the transpose of matrix A.
MAT C = (k)*A	Set matrix C equal to matrix A multiplied by the number k which must be in parentheses and may be given by a formula.
MAT C = INV(A)	Set matrix C equal to the inverse of matrix A.

Special MAT Functions. . . .

DET	Equals determinant of a matrix after inversion.
NUM	Equals number of components following a MAT INPUT.

Table 4: Matrix Special Functions of BASIC. The addition of these matrix functions to BASIC is one of the most common extensions to the language's capability. For engineering and scientific applications, such functions are a virtual necessity.

random way such as a dice roll in a game. The general form of the random function reference is:

RND

with no argument needed. If a particular BASIC implementation does require an argument, the number one is typically used:

RND(1)

To generate a random single digit integer, the following instruction might be used:

40 LET X=INT(10*RND)

If it is desired to generate a random integer number between two limits, this instruction may be used:

60 LET X=INT(A*RND+B)

This instruction will generate an integer random number between A and B, where A is larger than B. Thus, to simulate a dice roll for a game, with integers between 1 and 6, use:

INT(6*RND+1)

To aid in debugging programs, the typical BASIC RND function generates the same set of random numbers in the same order each time a program is run. In some implementations inserting a RANDOMIZE statement as the first statement in a program using random numbers, repeated runs of the program will produce different results.

User Defined Functions

In addition to the standard BASIC functions, some BASIC interpreters allow up to 26 additional functions to be defined with the DEF statement. The name of the defined function must be three letters, the first two always being "FN" (eg: FNA, FMB, ..., FNZ) and the DEF statement may appear anywhere in the program. Each DEF statement defines a single function and it can contain any combination of other functions and/or variables besides those denoting the arguments of the function. Any variable that is not an argument of a function will use its current value in the program. For example, to repeatedly use the function:

$e^{x^2} + 3X + Z$

define the function:

DEF FNE(X) = EXP(X**2)+3X+Z

and then call for different values of the function by:

FNE(2), FNE(A+B), etc

The current value of Z is used each time the function is called. Also, each defined function may have zero, one, two, or more numeric variables as arguments of the function.

Examples:

```
10 DEF FNA = 3.1.16*(R**2)
20 DEF FNB(X,Y) = (X+Y) / (X*Y)
30 DEF FNR(A,B,C,D) = FNB(A,B) +
   FNB(C,D)
```

Multiple line defined functions are constructed by dropping the equal sign and ending the function definition with a FNEND statement. The function name without arguments is used as a temporary variable to compute the function's value. Multiple line functions usually cannot be nested and there must not be a transfer from inside the function to outside its range, or vice versa. Also, GOSUB and RETURN statements are not typically allowed within a multiple line defined function.

```
Example:
10 DEF FNM(X,Y)
20 LET FNM = X
30 IF Y = X, THEN 50
40 LET FNM = Y
50 FNEND
```

This example returns the larger of the two argument values whenever the function is called.

ASC (single character or mnemonic of nonprintable character)
Returns a decimal value for the ASCII code of the argument.

CHR$(numeric formula)
Opposite function to ASC, argument is truncated to an integer which becomes the content of a one character string which is returned.

INSTR(numeric formula, string formula, string formula)
Searches for the second string within the first string. Search starts at character position specified by the numeric formula truncated to an integer or at the first character if omitted. Returns the position of the first character in the substring if found, 0 if not.

LEFT$(string formula, numeric formula)
Returns a substring of the string formula, starting from the left. Substring contains the number of characters specified by the numeric formula truncated to an integer.

LEN(string formula)
Returns the number of characters in its argument.

MID$(string formula, numeric formula, numeric formula)
Returns a substring of the string formula, starting at the character position specified by the first numeric formula truncated to an integer. The substring contains the number of characters specified by the second numeric formula truncated to an integer, or the substring continues to the end of the string if the second numeric formula is omitted.

RIGHT$(string formula, numeric formula)
Returns a substring of the string formula, starting from the right, containing the number of characters specified by the numeric formula truncated to an integer.

SPACE$(numeric formula)
Returns a string of spaces, the length specified by the numeric formula truncated to an integer.

STR$(numeric formula)
Returns a string representation of its argument (the number is converted to a string).

VAL(string formula)
The opposite function of STR$. Returns the number represented by the string formula (the string is converted to a number).

In addition to these functions, the CHANGE statement is used to convert between strings and numeric data and vice versa as follows:

CHANGE [string formula] TO [numeric formula]

Changes the string specified to a numeric vector. The zero element of the vector will contain the number of characters in the string and the decimal ASCII code of the letters will be stored in the numeric vector. The statement:

CHANGE [numeric vector] TO [string variable]

changes the numeric vector to a string. The zero element of the vector must contain the number of characters in the desired string.

Table 5: A "typical" set of string manipulation functions implemented in a BASIC interpreter. This table lists functions found in the author's experiences with BASIC on Digital Equipment Corporation computers; similar sets of string functions are often implemented by other "extended BASIC" interpreters.

Vectors and Matrices

Since operations on lists and tables occur frequently, a special set of matrix instructions and functions are usually available in BASIC. The list of typical MAT instructions and functions plus a brief description of their operation is found in table 4. Many of the matrix statements require special conditions to exist in order to be legal or may allow operations on vectors as well as matrices. While every vector has a zero component and every matrix has a zero column and row, these are ignored by MAT instructions. Any numeric array referenced in a MAT statement other than a MAT INPUT, will be set up as a matrix unless specifically declared as a vector in a DIM statement.

Strings

So far, only operations dealing with numeric information have been discussed; but some forms of BASIC also process alphanumeric information in the form of strings. A string is a sequence of characters including letters, digits, spaces, or some other printing characters. In BASIC it cannot, however, contain a line terminator such as a line feed, carriage return, form feed, or vertical tab. Strings are normally enclosed in quotes (eg: "ANSWER") when used as string constants, but may be omitted in some instances.

Variables may be used for simple strings and string vectors but not for matrices. Any variable or vector name followed by a dollar sign ($) stands for a string:

A$, B2$, or C$(1)

String variables or constants are used just as numeric quantities in most BASIC statements. However, numeric and string data are kept in two separate data blocks and are utilized independently of each other. The RESTORE statement discussed earlier resets both data pointers to the beginning of the data blocks. A typical implementation trick to reset only the numeric data pointer is to use: RESTORE*, and use RESTORE$ to reset only the string data pointer.

Table 5 shows some of the possible functions for manipulating strings in BASIC in some implementations.

Other Features

Many BASIC software systems, especially large time sharing systems, sometimes allow special operations involving data files and special formating of data. An example is the PEEK and POKE facility which is used in Altair BASIC for bit level manipulation. It is

always best to refer to the BASIC programming manual for the particular implementation being used as I have pointed out many times before. Always check what features are offered and how to use them, since they can vary drastically from system to system. There may also be special editing or operating commands available depending, again, on the system used.

In Closing . . .

Now that the general statements and functions have been discussed and some of the more advanced features have been introduced, the best way to actually learn the BASIC language is by hands-on experience. Find a system with BASIC, check the available BASIC features, then start with small, simple programs. In no time at all you should be able to program almost anything desired. BASIC is an extremely easy language to learn and use so don't let the term

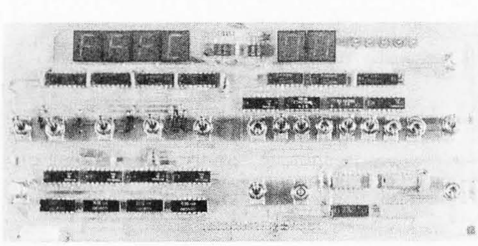
"HIGH LEVEL LANGUAGE"

frighten you away. Look for future articles on programming techniques, games, and applications as BASIC becomes available for more and more microcomputer systems. BASIC programs should be very easy to reproduce and convert for various specific systems, including microcomputers.■

What's New?

A New Microcomputer Series from CGRS

CGRS Microtech Inc, POB 368, Southhampton PA 18966, has announced a new line of microcomputers called the 6000 series which are all Altair bus compatible and based on the Mos Technology 6502 microprocessor. The Microputer I is a 3 board tutorial system featuring 256 bytes of programmable memory for $229.95. The Microputer II includes 2 K bytes of programmable memory with room for up to 4 K bytes of erasable read only

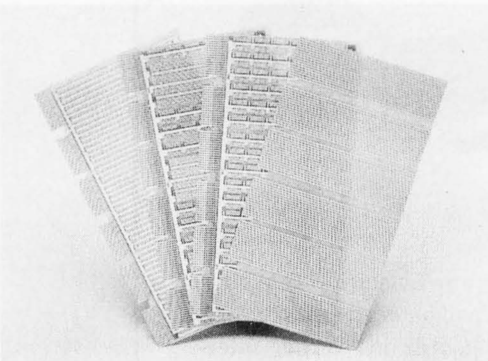


Circle 35 on inquiry card.

memory. The price is $389.95. The most sophisticated system in the series is the Microputer III, a complete microcomputer system upwards compatible with the first two systems. Contact the manufacturer for further information.■

Circle 572 on inquiry card.

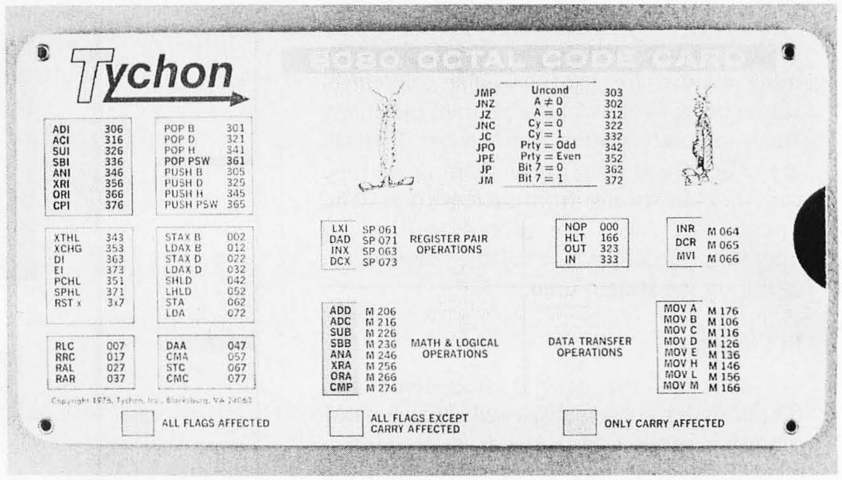
Breadboards Available for PDP-11 and PDP-8/A Users



Douglas Electronics has announced the availability of a complete line of breadboards designed to be compatible with Digital Equipment Corporation's PDP-11 and 8/A minicomputers.

The glass epoxy boards, which feature electroplated solder circuits and gold connector tabs, come in a variety of configurations and options which include extender boards with sockets, wire wrap boards and breadboards for custom logic circuitry. Contact Douglas Electronics Inc, 718 Marina Blvd, San Leandro CA 94577.

Circle 573 on inquiry card.



Attention Individuals, Clubs and Computer Stores: An 8080 Octal Code Card Is Now Available

Tychon Inc has announced this new 8080 Octal Code Card. The code card is a slide rule like aid for programming and debugging 8080 software in octal notation. It contains all the mnemonics and their corresponding octal codes. The instructions are all color coded to indicate which flags are affected during execution. The pocket sized card measures 6.5 by 3 inches (16.5 by 8 cm) and it provides the instructions in a neat, logical format for quick reference.

The back side of the card is printed with an ASCII code chart for all 128 characters plus the 8080 status word and register pair codes.

Delivery of the 8080 Octal Code Card is immediate and the price is $2.95 postpaid. Quantity discounts start at ten units, and custom imprinting is also available. According to Tychon, a hexadecimal version of the card will be available in the near future.

For further information write Tychon Inc, POB 242, Blacksburg VA 24060.■

Circle 574 on inquiry card.

A New Way to Do Slit-N-Wrap

Vector Electronic Company, 12460 Gladstone Av, Sylmar CA 91342, has just sent along this photo of a neat new addition to their line of Slit-N-Wrap tools, the P160-4T1 pistol gripped



110 VAC powered wrapping tool. This $80 tool will accept the P180 Slit-N-Wrap tool bit, or a standard wire wrapping attachment in the form of the P160-2A wrapping bit. While Slit-N-Wrap can be done by hand, it becomes much more convenient when it is powered as with this tool. (Because the Slit-N-Wrap technique requires no stripping of wire, it tends to be somewhat faster than standard wire wrap as a matter of course,

since the tool is always held in one's hand and is never put down for the stripping operation as in wire wrap with reel wire.)■

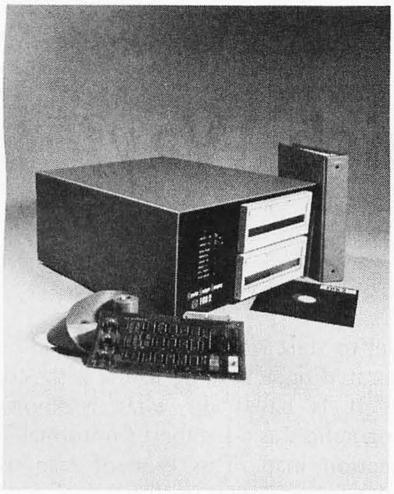
Circle 575 on inquiry card.

IMSAI Brochure and Catalogue

A beautiful four color brochure outlining the IMSAI product family came our way recently. In the same package was the current price list of this family of computers and computer products, which may be inspected at many local computer stores. A full catalog is also available according to the brochure, at a price of $1. Write IMSAI Manufacturing Corporation, 14860 Wickes Blvd, San Leandro CA 94577.■

Circle 576 on inquiry card.

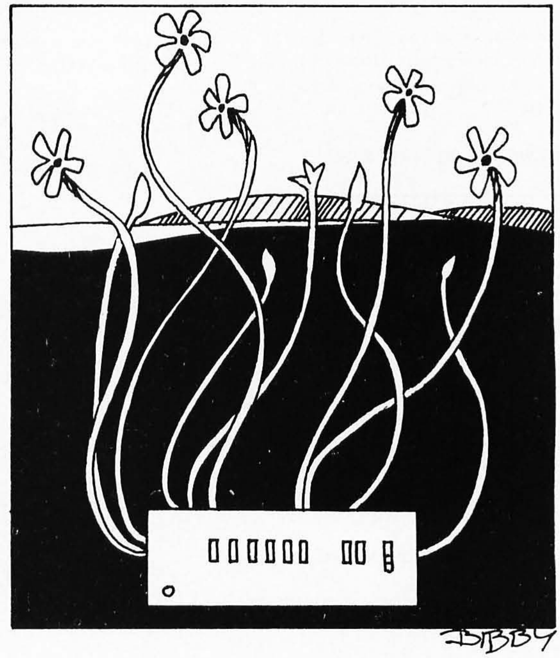
Altair Bus Floppy



Synetic Designs Company, POB 2627, Pomona CA 91766, has sent along this picture of their new "ready to use" disk system for Altair compatible 8080 based computers. The product is called the FDS-2, and it employs iCOM standard size floppy disk drives. The total package includes the iCOM executive system, text editor and assembler for the 8080, plus the custom "executive handler" provided by SDC to allow user-specific attributes of the executive. From one to four drives may be attached to the controller, with two drives shown in this photograph. Maximum on line capacity with four drives is just over 1 Mb. The recording format is IBM 3540 and 3740 compatible to allow transfer of data to and from other computer systems on floppy disk cartridges. Files may contain source data, program object data or user generated data. The drive may be seen at Computer Mart of New Jersey and other stores nationwide. Delivery is quoted as three weeks ARO for small orders; price was not quoted in the press release materials.■

Circle 577 on inquiry card.

Art
by
Duane
Bibby



How Far—Which Way?

Rene E Pittet
WMU Physics Dept
Kalamazoo MI 49008

Assisted by M Davidson

Did you know that your computer can answer the questions of how far and which way for you in microseconds for macrodistances? At least my SWTPC 6800 can with its 8 K BASIC high level compiler language.

The usefulness of this program will be apparent to anyone who sails a boat or flies an airplane. By inputting longitude (degrees and minutes) and latitude of the starting point first, and the destination second, the program will print out the distance in statute miles, or nautical miles with a minor change, and the true bearing. Look up the variation of magnetic north in your area and add that correction for a magnetic compass heading; or better yet, add it into the program.

The program of listing 1 is offered as a model to start from and can be modified or elaborated on according to the wishes of the user. It is based on what is known to cartographers as a Lambert Conformal Conic Projection map. This type of map uses a fixed reference point in its projection, so the method is less accurate the farther north or south you go, into Canada or Alaska for example, from the reference point in the US. However, it will be quite accurate within a restricted area such as that covered by an FAA aeronautical regional or sectional chart.

The requirement for this program grew from my Civil Air Patrol activities when we search for downed aircraft (a true hardware crash) and send rescue teams out to find them. Usually we have an airport to start from and grids to search in by air. The downed aircraft is spotted visually or by reception of a signal from its emergency locator transmitter (ELT). If found, its coordinates are radioed back to the mission coordinator. He then dispatches the ground teams to the rescue and here is where the questions "how far?" and "which way?" are answered by the computer.

The routine uses the trigonometric solution of a right triangle to determine the distance (hypotenuse) and direction (arcsine

$$a)\ \text{ARC SIN}\, x = x + \frac{1}{2 \times 3} x^3 + \frac{1 \times 3}{2 \times 4 \times 5} x^5 + \frac{1 \times 3 \times 5}{2 \times 4 \times 6 \times 7} x^7 + \frac{1 \times 3 \times 5 \times 7}{2 \times 4 \times 6} x^9$$

$$b)\ \text{ARC SIN}\, x = x + \frac{x^3}{6} + \frac{5 \times x^5}{40} + \frac{15 \times x^7}{336} + \frac{105 \times x^9}{3456} \quad \text{radians}$$

Figure 1: A series expansion of the ARCSIN function, as found (above) and as evaluated partially for use in the expression in listing 1 at lines 158 and 160 of the program.

of X). The sides of the triangle X5 and Y5 are the differences between the two sets of coordinates of longitude and latitude.

To explain the program a little, notice that the ARC SIN X is a special subroutine which had to be created since ARC SIN, ARC COS and ARC TAN functions are not included in SWTPC 8 K BASIC. This ARC SIN X function is performed by use of the trancendental formula shown in figure 1a which is reduced to the equation of figure 1b. The formula, translated into BASIC, is shown on lines 158 and 160 of listing 1..

Since the ARC SIN X formula above is less accurate from 45° to 90°, we switch and use 90° – COS of the complimentary angle of the right triangle and get the same resolution as when the angle is 0° to 45°. This is written in BASIC in lines 155, 163 and 168 to 175. Also, notice that in lines 58 and 59 a very small number (1.E-9) was added to the two sides X5 and Y5 of the right triangle X5, Y5, D. This extremely small error eliminates the problem of dividing into zero at the 90°, 180°, 270° and 360° points of the compass when one or the other of these sides of the triangle are zero. SWTPC 8 K BASIC doesn't like zeros in your math formulas.

In line 60 the hypotenuse of the triangle, D, is multiplied by 1.1516 to get statute miles. This conversion factor assumes that your calculations have been in British nautical miles which have 6080 feet to the mile. If your calculations have been in American nautical miles, with 6076.115 feet to a mile, the conversion factor would be 1.1508.

In lines 50 and 54 the numbers 44.89 and 74.82 were derived from the ratio of nautical British miles per degree of longitude versus latitude in the northern hemisphere at about 43° north latitude. In other parts of the globe, farther north or south, it would be advisable to recalculate to maintain accuracy. (The distance of one degree of longitude gets smaller and smaller as you go north or south from the equator.)

To correct for the magnetic variation in your area add the degrees of correction at lines 162 and 174. For example, if the aeronautical chart shows 3° west variation, then add +3 to R5's value.

One last comment: the latitude and longitude are entered as a decimal number. The digits to the left of the decimal are degrees and to the right, minutes; seconds are not used. The program sorts it all out and understands what you are inputting by using the INTEGER functions in lines 50 to 56.

Try it for aircraft flight planning or a sailing trip in large bodies of water, then check the accuracy by actual measurement on an aeronautical chart.■

```
00 PRINT   "-----------------------------"
05 LINE = 64
10 PRINT "TO COMPUTE DISTANCE & DIRECTION"
12 PRINT
14 PRINT "ENTER LONGITUDE (XX.XX) AND LATITUDE (YY.YY) OF BOTH PLACES:"
16 PRINT
18 PRINT "ENTER X1, Y1"
20 INPUT X1, Y1
22 PRINT
24 PRINT "ENTER X2, Y2"
26 INPUT X2, Y2
27 DIGITS = 2
28 GOSUB 50
36 PRINT
38 PRINT "-----------------------------"
42 PRINT
44 PRINT "TRY ANOTHER SET---------"
46 PRINT
47 GOTO 18
48 STOP
50 X3 = 44.89*INT(X1)+74.82*(X1-INT(X1))
52 Y3 = 60*INT(Y1)+100*(Y1-INT(Y1))
54 X4 = 44.89*INT(X2)+74.82*(X2-INT(X2))
56 Y4 = 60*INT(Y2)+100*(Y2-INT(Y2))
58 X5 = (X3-X4)*SGN(X3-X4)+1.E-9
59 Y5 = (Y3-Y4)*SGN(Y3-Y4)+1.E-9
60 D  = 1.1516*SQR(X5↑2+Y5↑2)
61 PRINT
62 GOSUB 150
63 PRINT "DISTANCE = ";D
64 PRINT "(STATUTE MILES)"
65 PRINT
68 IF X4>X3 THEN 74
70 IF Y4>Y3 THEN 84
72 GOTO 88
74 IF Y4 Y3 THEN 80
76 PRINT "TRUE BEARING = ";270-R5;" DEG'S"
78 RETURN
80 PRINT "TRUE BEARING = ";270+R5;" DEG'S"
82 RETURN
84 PRINT "TRUE BEARING = ";90-R5;" DEG'S"
86 RETURN
88 PRINT "TRUE BEARING = ";90+R5;" DEG'S"
90 RETURN

150 REM ARC SGN X SUB-ROUTINE
154 S = Y5/D
155 C = X5/D
158 S1 = S+(S↑3/6)+(3*S↑5/40)+(15*S↑7/336)
160 S2 = 105*S↑9/3456
162 R5 = (S1+S2)/.017453293
163 IF R5>45 THEN 168
164 PRINT
166 RETURN
168 C1 = C+(C↑3/6)+(3*C 5/40)
170 C2 = (15*C↑7/336)+(105*C↑9/3456)
172 R4 = (C1+C2)/.017453293
174 R5 = 90-R4
175 GOTO 164
```

```
------------------------------------------------------------
TO COMPUTE DISTANCE & DIRECTION

ENTER LONGITUDE (XX.XX) AND LATI
TUDE (YY.YY) OF BOTH PLACES

ENTER X1, Y1
? 83.00
? 42.24 DETROIT CITY AIRPORT

ENTER X2, Y2
? 87.55
? 41.58 CHICAGO O'HARE

DISTANCE = 255.92
(STATUTE MILES)

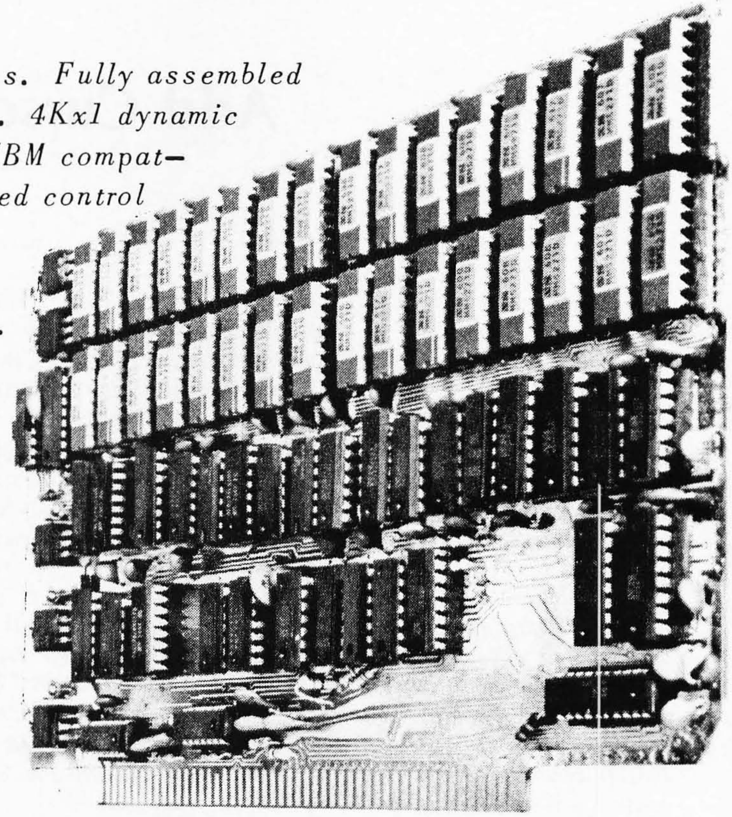
TRUE BEARING = 264.16 DEG'S
------------------------------------------------------------
TRY ANOTHER SET--------------------------------

ENTER X1, Y1
?
```

Listing 1: A BASIC program used to calculate range and bearing information from navigational coordinates. (This program was typeset from a hand typed listing of the original, which had been developed and debugged from a computer sans hard copy.) In the sample printout, user input is preceded by a question mark query.

16384 BYTES for $485.00

assembled (with sockets) : tested – burned-in – guaranteed



A new high in S100 bus memory cost effectiveness. Fully assembled (with sockets), tested, burned–in and guaranteed. 4Kx1 dynamic memory chips (the same ones used by the ton in IBM compatible memory systems) combined with self contained control logic, yield a memory system with:

- *Low power consumption, total board 5 watts.*
- *Transparent refresh, which means the memory looks static to the outside world.*
- *No waiting. In fact, xrdy is not even connected to the memory.*
- *Full DMA capability*
- *Reliable, low level clock and control signals.*

Dynabyte

Dynabyte® brings to the S–100 Bus a state of the art, industrial quality memory system. 16K on a single board for $485.00, Guaranteed for 1 year.

for more information call or write to:

R.H.S. MARKETING
2233 El Camino Real
Palo Alto, California 94306
(415) 321–6639

DEALER INQUIRES INVITED

B of A & MASTERCHARGE ACCEPTED

BYTE OF PASADENA
496 So. Lake Ave.
Pasadena, Ca. 91109
(213) 684–3311

BYTE OF SANTA CLARA
3400 El Camino Real
Santa Clara, Ca. 95051
(408) 249–4221

BYTE OF WALNUT CREEK
2989 N. Main St.
Walnut Creek, Ca. 94596
(415) 933–6252

BYTE OF SAN MATEO
1200 W. Hillsdale Blvd.
San Mateo, Ca. 94403
(415) 341–4200

BYTE OF PALO ALTO
2227 El Camino
Palo Alto, Ca. 94306
(415) 327–8080

Circle 37 on inquiry card.

Add Cursor Control to Your TVT II

Brother Thomas McGahee
The Salesian Center
80 S Sixth St
Columbus OH 43215

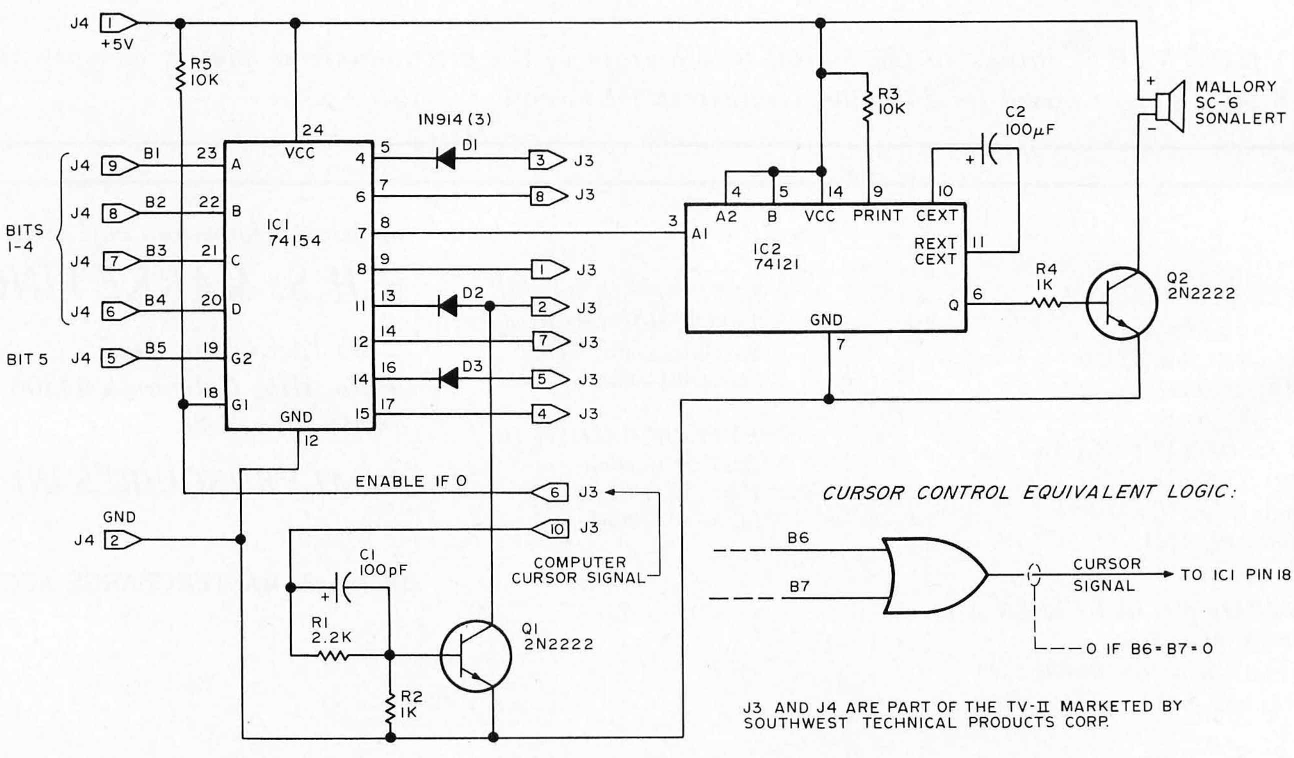
If you own a TVT II and would like a simple circuit to provide for operation of the cursor and erase functions, then consider the circuit shown in figure 1. This simple circuit will accept ASCII control characters and decode them in a manner suitable for controlling all of the cursor control and erase functions, and it even has a few outputs that can be used to provide user defined functions, such as enabling external devices or turning on bells and whistles.

In the version presented here several functions are provided, including home, erase to end of line, erase to end of frame, cursor up, cursor down, cursor left, cursor right, and bell. In addition there are five other unused codes that may be assigned any way you wish. Control J and M are unused, since the TVT II already decodes these as carriage return and line feed.

How it Works

IC1 decodes the incoming ASCII character, and selects one of its 16 outputs to go to logical zero whenever bits 5, 6 and 7 are all logical zero. The TVT II data lines are

Figure 1: Diagram for the TVT II cursor controller. IC1 is a 4 to 16 line decoder that will select which control command is wanted. Transistor Q1 is in the circuit to make it compatible to the Southwest Technical Products' screen read and UART boards. Diodes D1, D2 and D3 are in the circuit to protect IC1 when it is connected in parallel to the output of a gate on the main TVT II board. IC2 is wired as a one shot to activate the Sonalert when a control G is pressed.



Binary Code	ASCII Keys	Function	IC1 74154 pin #	How To Remember
0000000	Null	X	1	
0000001	Ctrl A	X	2	
0000010	Ctrl B	X	3	
0000011	Ctrl C	X	4	
0000100	Ctrl D	Down Cursor	5	D and Down are easily mnemonically related.
0000101	Ctrl E	X	6	
0000110	Ctrl F	Erase to end of Frame	7	F and Erase EOF are easily mnemonically related.
0000111	Ctrl G	Bell	8	G is the standard bell function in ASCII.
0001000	Ctrl H	Home	9	H and Home are easily mnemonically related.
0001001	Ctrl I	X	10	
0001010	Ctrl J	Line Feed	11	
0001011	Ctrl K	Right Cursor	13	K was chosen for Right not so much because it was desirable so much as the fact that it was the least undesirable. A, B and C are commonly used program control commands. J and M are used for carriage return and line feed. This leaves us with only E, I or K. I chose K and use the mnemonic K for Kontinue.
0001100	Ctrl L	Erase to end of Line	14	L and Erase EOL are easily mnemonically related.
0001101	Ctrl M	Carriage Return	15	
0001110	Ctrl N	Up Cursor	16	While N is not mnemonically related to Up, it was chosen since many keyboards have an upward pointing arrow above the N.
0001111	Ctrl O	Left Cursor	17	O is not mnemonically related to Left but was chosen because many keyboards have a left pointing arrow above the O.

Table 1: This table summarizes the inputs to and the outputs from IC1. Along with the desired keycode is given the binary input to IC1, the output pin that is enabled, and the function that it is wired for according to the circuit of figure 1. At the right is an explanation of the alphabetic mnemonic that is used to remember the correct key to cursor command code. Note that with the exception of the bell, line feed and carriage return commands, these functions are not standard ASCII functions.

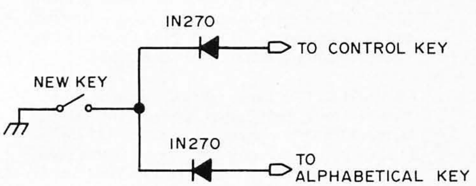


Figure 2: Simple circuit to add control keys to a keyboard. The new key is wired to both the control key and the alphabetical key. The diodes are needed to isolate the keys from each other.

labelled from 1 to 7 by SWTPC. Bit 5 is routed to IC1's G2 input, and the G1 input of IC1 is fed from the TVT II "computer cursor" signal, which is zero whenever bits 6 and 7 are both logical 0. Bits 1 to 4 are applied to the A, B, C and D inputs of IC1. Whenever one of the first 16 control characters is detected, one of the decoder's outputs will go to logical zero.

You will notice that some of the outputs are connected to diodes. This is to protect the output of IC1 when it is connected in parallel with the output of a gate on the main TVT II board. You could use an open collector version of this decoder integrated circuit, but I chose to use a 74154 because they are readily available at low cost. I have 1N914 diodes on my unit, and have not had any problems. A germanium type, such as a 1N270 might be a better choice, but I simply used whatever I had on hand.

Q1 is included in the circuit so that this circuit will be compatible with SWTPC's screen read and UART boards. It provides logic to advance the cursor when told to do so by the other boards. IC2 is a one shot set up to handle the bell function. A control G will cause a low going pulse at its A1 input. It will then stretch the pulse for a second or so to drive the Sonalert, or whatever you wish to use as a bell function. There are several other outputs from IC1 that can be used to initiate other bells and whistles in a similar manner. The J numbers in figure 1 refer to the connector plugs on the main TVT II board.

Table 1 is a listing of the 16 outputs and their function assignments. An "X" in the table indicates a free function waiting to be implemented. A mnemonic explanation of the functions at the right of table 1 helps when using the keyboard. If for any reason you wish to change which control keys perform which functions, just wire to a different output pin on IC1.

If you are using a diode encoded keyboard, you can add cursor control and erase keys quite easily, if you already have a control key. All you do is connect each cursor or erase key as shown in figure 2. The diodes serve to isolate the keys. Whenever the key is depressed it will simultaneously activate both the control and alphabetic keys. You can add as many of these type keys as you need. While you are at it, you may wish to add a calculator type array of switches for numerical entry. You simply wire them in parallel with existing numerical keys; no diodes needed. This will considerably speed up the entry of numerical data.■

Classified Ads

WILL TRADE: AKC Samoyed puppy or stud service for computer (or related) hardware, BASIC software, or texts of interest. Contact G Frank Humiston, 506 Wayne Av, El Cajon CA 92021.

FOR SALE: Complete Wang computer system. CPU (Model 2200A) contains 12 K RAM expandable to 32 K. BASIC is resident in 20 K ROM. CRT/tape drive (Model 2216/2217) in one console. 12 inch diagonal 64 character by 16 lines screen. Drive is fully under program control. Alphanumeric keyboard (Model 2222) with separate numeric key pad and 32 special function keys. Commercial quality equipment excellent for small business application. All for $4,500 or trade (new at $9,300). Write or call for data sheet. Al Leung, POB 5272, Redwood City CA 94063, (415) 364-6746.

HELP: I found some partially assembled electronic equipment and other items for Four-Phase Systems Inc sync modem monitor two summers ago in the Flaming Gorge National Recreation Area UT. I have been unsuccessful in tracing the owner. To claim, let the owner identify the equipment in full. Jay Radavich, 120½ S Polk, Moscow ID 83843.

FOR SALE: MOS Technology Inc KIM-1 with 1 K homebrew 21L02 RAM. Power supply: three each +5 V at 1.5 A, 1 each –5 V at 0.5 A, 1 each at +12 V 0.5 A, 1 each at –12 V 0.5 A. Serial adaptor board, SAD-1 (checked out on a CT 1024, works OK), wood case, extra 22 pin connectors with two extra on case, cross assembler manual, TIM manual. Please monitor with tape and books, Tiny BASIC with tape and hexadecimal list, 10 each Memorex, 30 minutes each side with programs. $330 takes all. Robert G Lloyd, 7554 Southgate Rd, Fayetteville NC 28304, (919) 867-5822.

WANTED: To buy, for personal use only. Tested programs for business applications, business games and simulations. Micro BASIC or IBM 360 BASIC/FORTRAN IV. Doug Gilson, #72 University Ter, Columbia SC 29201.

WANTED: Young student with small budget wants back issues of BYTE, *Radio Electronics, Popular Electronics*, and computer and electronics books (at list price or less plus postage). Will buy or trade with back issues of *Popular Science* or *Popular Mechanics.* (I have them back to 1932.) Contact me at the next meeting of SDCS or write Victor O'Rear, POB 1388, La Mesa CA 92041.

FOR SALE: IMSAI one UCRI cassette board $39, one 8080 processor board $160, one 4 port parallel IO board $140, one 2 port serial IO board $140, two parallel board cables $20 each, two serial board cables $15 each, one SWTPC PR-40 printer $240, one Oliver Audio paper tape reader $60, one DC-50 MHz RF generator $200, one Motorola PT-300 Handi Talkie $250, one Software by Scelbi SCELBAL higher level language $35, one 8080 assembler $20. All boards fully assembled and guaranteed to work. Will ship COD UPS; you pay the freight. Vance James, POB 2292, Hickory NC 28601.

Readers who have equipment, software or other items to buy, sell or swap should send in a clearly typed notice to that effect. To be considered for publication, an advertisement should be clearly noncommercial, typed double spaced on plain white paper, and include complete name and address information. These notices are free of charge and will be printed one time only on a space available basis. Insertions should be limited to 100 words or less. Notices can be accepted from individuals or bona fide computer users clubs only. We can engage in no correspondence on these and your confirmation of placement is appearance in an issue of BYTE.

Please note that it may take three or four months for an ad to appear in the magazine.■

NEEDED: BYTE issues volume 1, numbers 1 to 10. Must be complete, and preferably in at least good condition. High quality Xerox reproductions may be considered (if BYTE doesn't mind). For Xerox, please forward a sample reproduced page. T Mishler, 5914 Glenn Av, Flint MI 48505.

FOR SALE: CRT terminal, 3 month old Micro-Term ACT-1 with 9 inch Sanyo monitor. 16 lines of 64 characters, autoscroll, 110, 300, 600, 1200, 2400, 4800, 9600 bps, RS232C or current loop. Modified for direct display of typed data. All documentation. $495 includes shipping. Bob Ripley, Rt 3, Fayetteville AR 72701. (501) 267-3172 evenings.

FOR SALE: To highest bidder, all issues of BYTE magazine published to date, in mint condition. Will not break up collection. Only highest bidder will be notified. Send offers to David Hicks, 1711 E Willow Ave, Wheaton IL 60187.

WANTED: Technical info on SR-52. Labels, format and signal levels of interconnection pins to printer, card reader data format and recording technique, IC pinouts, schematic, programming, architecture, signal levels, voltages, anything that will help me interface it with a homebrewing Z-80 and/or reload program from an outside memory. Will pay duplication costs or will buy a service manual if available. Will work or exchange information with anyone who has attempted to interface this unit. Help. Jack Lambert, 5 Graham Rd, Lexington MA 02173, (617) 862-8490.

BYTE: Issues 1 thru 13 in good condition, complete set for $39. Tom French, 404 Leeward Lndg, Roswell GA 30076, (404) 993-1042.

WANTED: Back issues of BYTE. Need September, October, November, December 1975 and April 1976. Also need December 1975 and January, February, March 1976 of *SCCS Interface.* Send your asking price to Dick De Nicola, 516 Devanah St, Covina CA 91722, (213) 685-6307.

FOR SALE: Viatron 2111 Data Management Center with Printing Robot and all manuals. Unit is in operating condition, $700. Dan Wolf, c/o Interglobal Photos, New York NY 10003.

FOR SALE: Southwest Technical Products TV typewriter. Includes keyboard, 1 K of RAM, serial interface board, power supply, cursor controls and two page memory options wired. Mounted on a metal chassis. Complete wired and in working condition, $200. Webber Hall, 16967 Blanche Pl, Granada Hills CA 91344, (213) 363-2004.

FOR SALE: 7 track AMPEX FR400 Digital Tape Handler; includes all motors, motor control circuitry, hubs, capstan, take up reel for 2400 foot 1/2 inch tape, complete documentation and parts list. No read write electronics or manual control panel. Best offer plus shipping for 160 lbs, about 19 by 17 by 17 inches. Paul Krieger, 2111 W Iliff, Englewood CO 80110, (303) 922-2385.

FOR SALE: Paper tape equipment. Two high speed Digitronics 8 level paper tape readers 300 cps, stop on character, like new, $60 each. One 8 level paper tape punch, including all drive electronics and interface to 8 bit micro, $75. Jim McCord, 120 E De La Guerra, Santa Barbara CA 93101, (805) 963-8941.

FOR SALE OR TRADE for working or nonworking mini computer. SCR 826MA, Horizon 2, Lampkin 105B, etc. Please write Dale Hutchinson, 10818 Brentway Dr, Houston TX 77070.

FOR SALE: One 6502 KIM system with documentation, $175. One 6502 MPU and 6530-004 TIM with TIM manual, $45. Three 6530-005 RAM, IO port combinations, $10 each. Eight 2102s, $12 the lot, and two 6111A-4s, $6 for both. Miscellaneous TTLs: 74100, 74150, 8202, 8416, $.50 each. I'm house cleaning and hope someone can use these. J Grina, 1233 Ray Pl, St Paul MN 55108.

FOR SALE: SCM Typetronic 98/330 (CDC-9816) 2R/2P with Fron feed carriage. Includes control unit IO printer, processor, two readers and two V-punch united. Original cost $23,000. Best offer. Mr Hepburn, Rockland MA, (617) 871-1790.

FOR SALE: MITS Altair 8800a. Assembled, factory tested. Includes: CPU card with 8080a, factory assembled SIO-C (TTY) interface card, ACR (cassette) interface card, factory adjusted to the new MITS frequencies, two 4 K dynamic memory cards, 12 total factory installed edge connectors and card guides, switchable fan and MITS modified power supply. Originally cost over $1400. Will sell for $850 (or make me an offer). Call Craig Pearce at (312) 484-5846 (7 to 10 PM, CST).

FOR SALE: Complete factory assembled Sphere system: 6800 MPU, keyboard, CRT, PWR supply, resident monitor in PROM, 4 K user memory, 2 cassette IO. Asking $1000, or willing to trade for a working IBM 1620. Bob Wada, 12812 Court St, Garden Grove CA 92641, (714) 893-8940.

WANTED: System Viatron 21 and accessories. Fred De Bros, 15 Garden St, Boston MA 02114.

WANTED: I would like to purchase the first 12 issues of BYTE (ie: volume 1, numbers 1 to 12). If you have any of these issues for sale please write W R Parks, Assistant Professor of Computer Science, Walters State Community College, Appalachian Hwy, Morristown TN 37814.

WANTED: Selectric typewriter/terminal compatible with Altair type system. Need either complete unit or conversion kit including interface for Selectric I or II. Charles Ames, 46 Coburn Woods, Nashua NH 03060, (603) 889-5000 after 7 PM.

FOR SALE: One MITS 4 K static memory board and one MITS 4 K dynamic memory board, $165 each. One Processor Tech 8 K memory board $315. All boards fully socketed, fully assembled, and work perfectly. J R Fish, 2121 Columbia Pike #208, Arlington VA 22204, (703) 521-6763.

FOR SALE: TTY paper tape, 1 inch wide, 8 inch diameter rolls, buff, oiled, perfect for ASR-33. $5 plus UPS shipping per 7 roll carton (12 lbs) or $15 per 28 roll case plus UPS. Two or more cases, $12.50 each plus UPS. Dan S Parker, 1007 3rd St #3, Davis CA 95616, (916) 758-2341 evenings.

FOR SALE: Burroughs Card Reader (Series A590, Style #A592), its logic unit, and a complete set of schematics. Any offer will be welcome. Its physical dimensions are 10 by 9 by 24 inches. Chris Nolet, 131 Bridge St, Dedham MA 02026, (617) 329-3189.

WANTED: Manual, schematic and service data for Burroughs calculator Model C-5155, Series C-5000. Luis Pena, POB 954, Montevideo, Uruguay, S AMERICA.

FOR SALE: 1702A by Intel. Bought large lot to cut expense. Willing to share savings. Factory fresh. $5.50 plus $1 for shipping and packaging. B D Lichtenwalner, 29 Michael Rd, Stamford CT 06903.

FOR SALE: First 24 issues of BYTE complete. Best offer. Roy Lynch, POB 8, Sabina OH 45169, (513) 584-2424.

FOR SALE: PDP-11/03, cheap; 16 K + 24 K board for same; portable TI terminal; twin floppies, RT-11. Make offer for all or any of the above. Also LA36 DECwriter II, $1200. Omnitech 701B modem, $190. Would like to discuss all DEC or DEC compatible hardware and software. Need PDP-11/04 or larger. Walter D'Ull, 2239 Grand Concourse, New York NY 10453, (212) 933-0300 or (201) 744-0685.

WANTED: Information on how to interface a Flexowriter Programatic to a processor. I will pay for cost of duplicating and postage. Have Technical Manual and schematics for Models FL, SFD and SPD for anyone who needs it. Frank Goeringer, 6 Fox Meadow Ln, Cahokia IL 62206, (618) 337-7031.

SOFTWARE WANTED: Bio-engineering or biomedical simulation programs for IMSAI 8080 in BASIC, PDP-11 in BASIC or FORTRAN, WANG 2200 in BASIC, IBM 360 or IBM 370 in almost any language. Donald Becnel, POB 14473, Baton Rouge LA 70808.

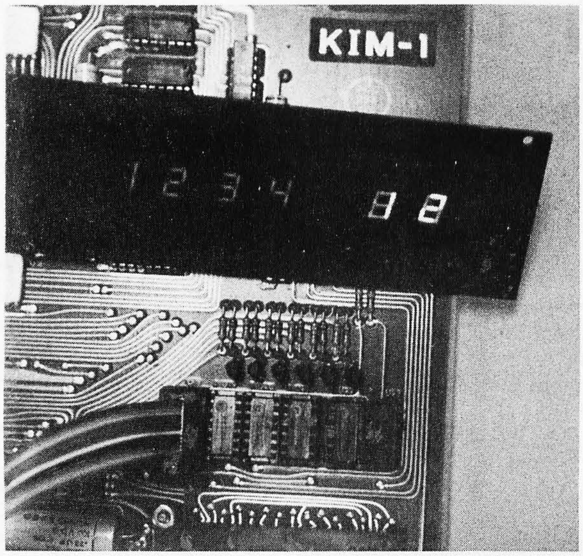


Photo 1: The homebrew remote display board, plugged into KIM via a ribbon cable. With such a remote display and a remote keyboard connected through KIM's edge connector, it is possible to create an enclosed housing which protects the processor.

Giving KIM Some Fancy Jewels

Robert Grater
1595-21 Laurelwood Rd
Santa Clara CA 95050

When my KIM-1 arrived I was excited, but a little dubious; after all, I had read all those microprocessor books and still didn't know beans about programming. Well, MOS Technology's excellent manuals took care of that and I'm on my way to knowing what I'm doing. When I say excellent, I mean if I learned it, anyone can!

But KIM has a fundamental problem! I like KIM, but I just can't see myself sitting there with a naked board gathering dust and me dropping ashes all over it while I sweat out a program. Also with MOS Technology's welcomed announcement of the KIM-2 and KIM-3 programmable memory boards and the coming KIM-4 mother board, it's really time to start thinking about putting KIM in a nice enclosure. But in order to bury KIM in an enclosure, I had to somehow bring the displays and keyboard lines out.

MOS Technology was nice enough to bring the keyboard out to the application connector, so one problem was solved before I started. But what about the display? I didn't want to just parallel the existing readouts on the board because that would mean that much extra current, and this might not make the drivers too happy. I came up with the following solution which is fairly simple, yet a bit delicate to start with. Delicacy and discretion with KIM are requirements of the task of unsoldering the original readouts,

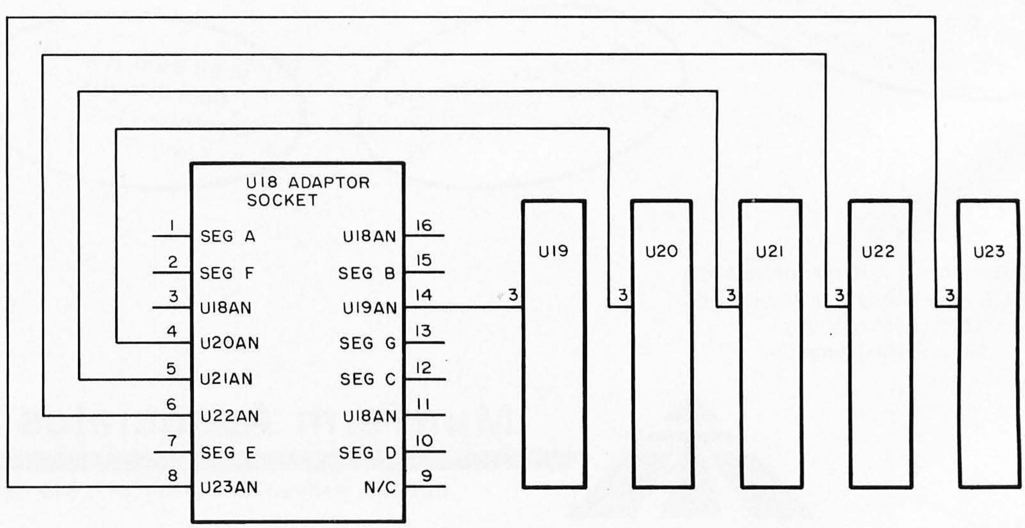


Figure 1: This diagram shows electrical connections of the 16 pin remote adapter socket installed as U18, with patches to the pin 3 anode connections of U19 thru U23 of the KIM-1 design. The display positions of U19 thru U23 can have 14 pin sockets installed if it is desired to occasionally revert to on board displays.

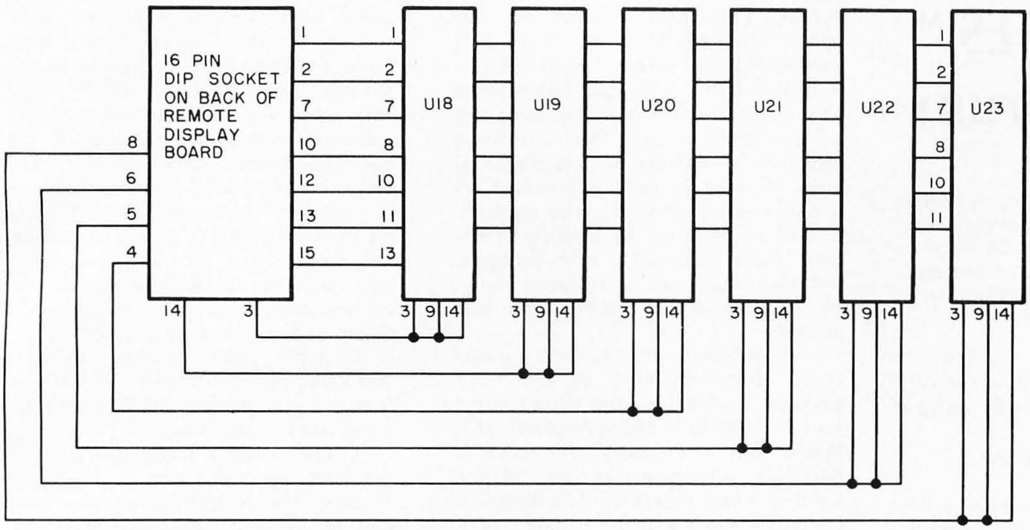


Figure 2: A remote display board. After removing the KIM-1 displays, a remote display board is constructed for mounting in a more convenient packaging arrangement. The wiring of the cable which plugs into the 16 pin socket installed at the U18 position of the KIM-1 board is listed in the diagram.

SUMMARY OF INTERFACE CABLE

Pin	Connection	Pin	Connection
1	Segment A Cathode	9	No Connection (Do not drill on KIM)
2	Segment F Cathode	10	Segment D Cathode
3	U18 Common Anode	11	U18 Common Anode
4	U20 Common Anode	12	Segment C Cathode
5	U21 Common Anode	13	Segment G Cathode
6	U22 Common Anode	14	U19 Common Anode
7	Segment E Cathode	15	Segment B Cathode
8	U23 Common Anode	16	U18 Common Anode

turning these displays into remarkable jewels.

The MAN 72 7 segment readouts (U18 thru U23 of the MOS Technology documentation) are unsoldered with an iron of less than 35 W and a "solder sucker" or similar vacuum solder removal tool. This is the only way I've ever found to successfully unglue items on double sided boards. This will allow you to replace the displays with one 16 pin socket as shown in figure 1. The 16 pin socket will go in the 14 pin U18 slot (there is room for the extra pin below). Use of the unused pins of the old 14 pin display pattern, plus one extra socket pin of the 16 pin socket, allows all the necessary control signals to be routed to the external display position.

In order to mount the 16 pin socket, take a small piece of Vectorboard with 0.1 inch spacing and lay it over the spot for U18 and line it up with the existing holes; this will be your marking guide. Now using a scribe or small drill, mark the spots for pins 4, 5, 8, and 14 (which were originally unused at U18). Do not mark the spot for pin 9 since it would go through the circuit on the other side. Hold the Vectorboard firmly when marking and check for alignment with a 16 pin socket after marking it. Now using a #58 drill at the highest speed you can get, carefully drill the extra four holes. Take your 16 pin dip socket and clip off pin 9 on the back side; now it may be soldered in place with pin 1 to the upper left. (If you solder five 14 pin sockets in the other display patterns, you can plug the six displays you removed into the sockets and check to make sure KIM is still talking to you).

Now we have four extra pins at U18 and five anode leads to bring out on U18's modified socket (which is called the adaptor socket) in figure 1. Actually we have five extra pins since pin 6 on all the displays is the decimal point cathode and is not used. All we have to do now is bring the five other common anode leads from U19 thru U23 to our modified U18 socket as illustrated in figure 1. This may be done with very small (#26) wire. I picked up pin 3 of all the other display sockets which seemed to give the easiest routing, straight down the middle. The wiring diagram of figure 1 shows the connections. Now you have the original on board display with the option of plugging into the adaptor socket with an 16 pin DIP plug for a front panel display when you put KIM in a box.

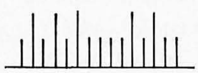
The front panel display may be wired as shown in figure 2. I used a modified clock display board and would have been just as well off wiring it by hand on Vectorboard. Also I used a 16 pin DIP male to male plug with another 16 pin socket mounted on the back of the display board, so the whole works can be unplugged if I want to go back to the bare KIM for testing or whatever. Just for a little added elegance and pizzazz, I replaced the two data display readouts with MAN 7G green LED displays for some classy emerald jewel effects.

This modification of KIM's display jewels is just the first step in getting KIM into a more workable medium for expansion and dressing her up a bit. I've also been working on the complete housing, but that's another story.■

PAPERBYTES™ Forum

Yet Another Format?

Ian Robinson, 308 W 5th St, Neillsville WI 54456, sent in an extended letter on an alternative to the bar code format we proposed in November and December, and have begun using. He proposes a format which looks like this:



— data bars (1s only)
— clock bars (each bit)

010101000010100   Binary data

However, there is one disadvantage to this format: It requires two read sensors since the data clock bars and the data bars must both be read simultaneously. He argues that this method can be hand drawn with quite a variation in density, which may be true, but the whole idea of the bar codes was to automate the *printing* process for mass produced software distributed in magazines. ■

BAR CODES AND OTHER TOPICS

I enjoyed the April 1977 issue of BYTE, which I thought was somewhat improved in areas of my interest. In particular, I enjoyed your article, "A Software Controlled 1200 bps Audio Tape Interface." For a while, I thought that our interests might be diverging, but now your articles are tending more toward the hardware direction, with one particular area of common interest that intrigues me greatly.

The idea of published software in bar code form could well be the greatest communications method yet, as far as hobby level exchanges go. Here are some additional ideas for your consideration.

An almost perfect version of the optical reader described by Frederick Merkowitz in his article on page 77 of December 1977 BYTE is available from Wilcox Enterprises, whose ad also appears on page 85 of this issue. This item sells for a mere $15, and contains a B & L Microscope equipped with an EPI illuminator (through the lens) and a 10 x objective lens. The microscope assembly is equipped with a photocell and slit assembly, and is mounted on a precision set of slides. The assembly is scanned or driven by a motor (every unit I have ordered has been damaged, but the motor can be replaced easily, by any semiskilled hacker) and gear assembly.

An infrared emitter and sensor combination, mounted in a TO-18 package, is available from Ultra Sensors Inc at 2400 West 102nd St, Suite 313, Minneapolis MN 55431. The price is $4.50 in unit quantities (ask for the Cyclops Evaluation Kit). This device is easily made into a light pen style reader, or any other configuration one might desire. There seem to be no significant problems with ambient light, or angle of approach with this device, if the self-contained lens is held in contact with the paper while reading. This device works best if it is powered by approximately 10 kHz pulses of 5 to 10 V magnitude, through a 27 Ohm current limiting resistor (or it can be driven by a single UJT transistor oscillator operating at 10 kHz). Connect the photocell to a LM 308 op amp through a 100 pF capacitor followed by a 3.3 k Ohm resistor to ground (ie: a differentiator). If you rectify the output of the op amp, you have the scanned bar code signal!

A bar code printer (coded for the standard bar codes, ala super market) is available from:

Medical Record Systems
Division of Ames Color-File Corporation
12 Park St
Somerville MA 02143

They ask a very high $6,500 (in hobby terms) price for this critter, and the electronics for bar code formulation and interface are not of particular interest in our case. Possibly the printer mechanism alone can be purchased. With such a machine connected to a keyboard and a universal (ie: programmable) cassette tape interface, BYTE magazine could reproduce almost anyone's software recordings into one of the bar code formats that you have been discussing. This would make the dream of software exchange a reality. (If BYTE would like a formal quotation for construction of such a system, just ask. This is my business. A first guess at the cost, with bar code printer, would be approximately $13,000.)

I don't enjoy writing software, and I find the manual entry of software listings a very boring task. Published bar codes could change this.

I have recently made several video monitors by converting a Magnavox 12 inch, transformerless set for a total cost of $87 each. I go to a Magnavox dealer, and purchase an MG 5056WAF2 less the video, audio and tuner modules. These are plug-in and can be removed in minutes right in the show room (take a 1/4 inch nut driver with you). The dealer can use these modules to service quite a few Magnavox sets, and the total rebate has been $53.55 to me. By sawing the fingers off an old PC card, you can fabricate a connector to fit into the video modules socket, and make connection to pin 4 V (video) and pin 8 (ground). Delta Electronic's L9971 isolation transformer at $6 then mounts in the same screw holes that mounted the tuners! If you think there would be sufficient interest in this project, I could put together a project article on same.

The front cover picture of April 1977 BYTE is fantastic. How about poster size reproductions of this? I need several (to justify the present conditions on several of my own benches)!

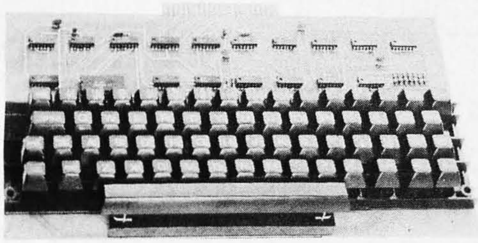
Sumner S Loomis
Route 1, Box 131 A
Prairie Point MS 39353

Thanks for your excellent comments and suggestions, but I thought I used software to generate and decode the information at the interface. . .CH ■

Circle 97 on inquiry card.

What's New?

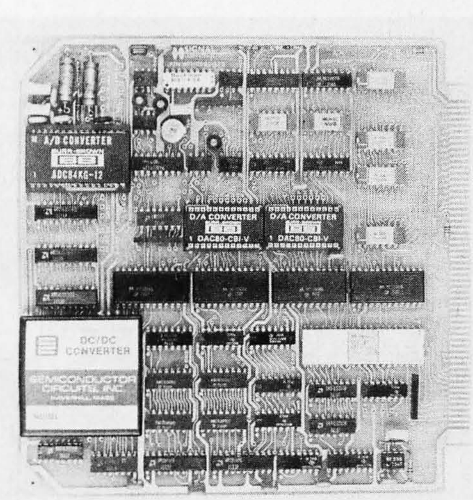
ASCII Keyboard Kit from Radio Shack



Radio Shack has announced the new Archer ASCII Keyboard Encoder kit for $57.80. The price includes all electronics and parts needed to assemble the keyboard as shown. An external 5 VDC 500 mA power supply (not supplied) is necessary to power the unit; this may be purchased locally. The printed circuit board alone, including assembly and parts manual, is available for $14.95. See your local Radio Shack dealer for details on their latest catalog sheets.■

Circle 634 on inquiry card.

A New Multiplexed Analog IO Card for the Z-80



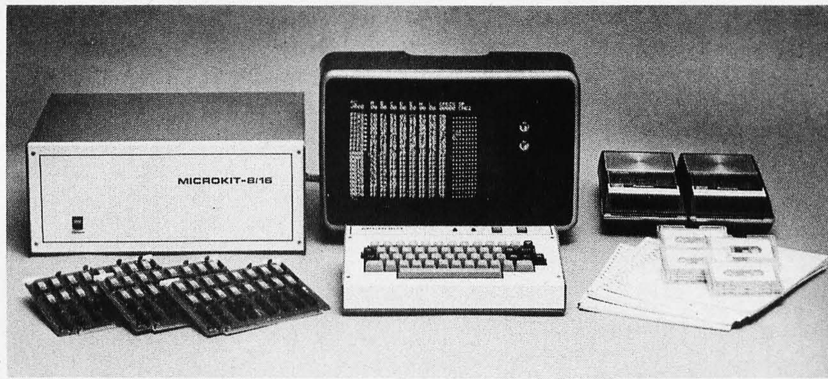
Signal Laboratories Inc, 202 N State College Blvd, Orange CA 92668, is making available a new analog IO card. With this board, intended for use with the Zilog Z-80 development systems, the engineer or scientist can monitor up to 16 analog channels in real time.

Circle 51 on inquiry card.

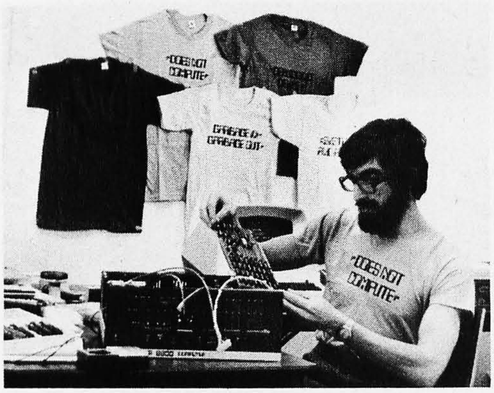
The channels are sequentially sampled and sent through a 12 bit analog to digital converter. Two multiplexed digital to analog outputs are also available on the board, as well as four discrete (1 bit) outputs and a DC to DC converter, enabling the board to be powered with only +5 VDC. One analog to digital conversion takes a maximum of 20 μs. A programmable gain amplifier is also included offering gains of 1, 2, 4 or 8. An option is available which allows the board to monitor up to 32 analog input channels. Price for quantities under ten ranges from $595 to $910, depending on options. Available from stock.■

Circle 645 on inquiry card.

A Sophisticated New Small System



Microkit Inc, 2180 Colorado Av, Santa Monica CA 90404, has announced their new Microkit 8/16 Universal Microcomputer Development System as shown for $5,275. The system is available in both 8080 or 6800 versions, and the price includes 32 K bytes of programmable memory, two cassette recorders and a high speed graphics display. An interesting feature of the software support package, called Quickrun, is that the monitor, debugger, editor and assembler reside in the 32 K byte memory simultaneously, along with a source code work space and an object code work space. This allows up to 1000 steps of source program and 4 K bytes of object code to reside in memory simultaneously.



On Line Shirtware

Fed up with those tiresome Paris originals? Eschew the tacky route at your next computer club meeting with Martha Herman's new computer T-shirt designs. Try REAL TIME OPERATOR on a dark green background, or maybe GARBAGE IN—GARBAGE OUT on orange. Other titles include COMPUTER WIDOW and RANDOM ACCESS; there are ten in all.

Shirts are available from Martha Herman, 114 W 17th St, New York NY 10011, for $5 plus $.60 postage per shirt. State chest size.■

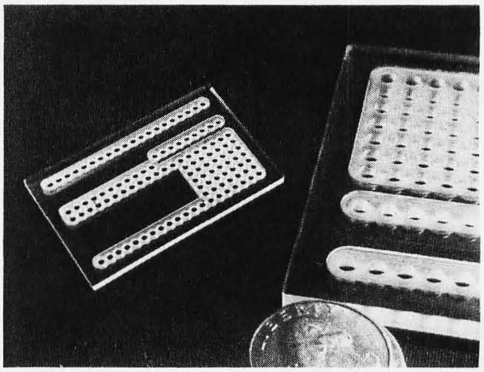
Circle 646 on inquiry card.

The user can thus switch at any time from editing to assembling to debugging.

Quickrun allows the use of symbolic debugging. Being able to refer to memory locations by their symbolic names eliminates the need to consult program listings to find hexadecimal memory addresses. Also included in the system are memory protect features and a microemulator. The latter allows the user to simulate up to 20 K bytes of memory.■

Circle 647 on inquiry card.

A New Drafting Aid for Printed Circuit Layouts

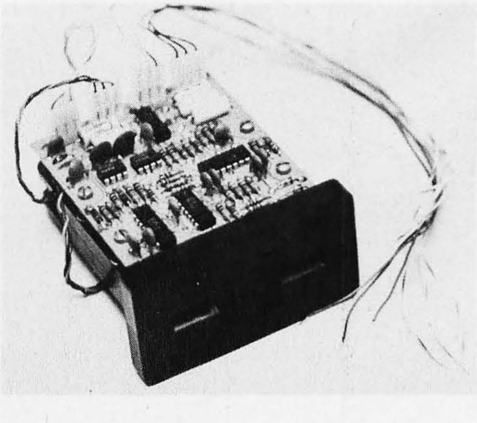


The Machine and Control Division of Interrotech, POB 128, Farmington ME 04938, has announced a new drafting template for $8.95 to be used in designing "tapeless" printed circuit board layouts using ink pens, or pencils. The template produces pin layouts of dual in line integrated circuits that are twice actual size. The template has a versatile pattern that includes all the popular sizes up to 40 pin packages.■

Circle 635 on inquiry card.

Badge Reader Module Announced

RD Products Inc, 6132 Route 96, Victor NY 14564, have announced their Mark I badge reader for $100 in quantities of 1 to 25. The reader is designed to read ID cards (such as credit cards) having magnetic stripes. A maximum of 12 decimal or hexadecimal digits can be read on a magnetic stripe having



Your microcomputer is only as good as its memory



Your microcomputer is... in a manner of speaking... capable of thinking. The amount of information you can store, combined with the access speed, determines the bottom line capability of your microcomputer.

The Midwest Scientific Instruments' FD-8 Floppy Disk Memory System will enable you to get the most out of your microcomputer...6800 or 8080. We have the most complete 6800 disk operating system available today, which interfaces to any microcomputer via a single PIA chip.

When used with the MSI FDOS Operating System...included with purchase...the FD-8 gives your microcomputer a level of performance which can only be matched by larger computers.

You can have program save and load with named, variable length, password protected files at three levels: object code files, source code files, and BASIC files.

It has disk routines which can be used easily by low level assembler language programs as well as high level BASIC programs.

The coresident assembler/editor saves and loads source files from disk, assembles and stores object code on disk, and creates assembly listings on a high speed printer.

You get a full ANSI standard BASIC interpreter* with program load and save, from disk as well as tape, with both random and sequential disk data files. BASIC also has multi-I/O port capability under software control.

And all you need do to access any of these functions is type the appropriate commands on your terminal.

The FD-8 is complete, including all power supplies, cabinet, cables, software, manuals, and documentation.

You can order the FD-8 as a kit for $1,150.00, or wired and tested for $1,395.00.

Visit one of our dealers for a demonstration, or send for our free Catalog of MSI products.

American Microprocessors
Equipment & Supply Corp.
Chicagoland Airport, P.O. Box 515
Prairie View, Illinois 60069
(312) 634-0076

Microcomputer Systems, Inc.
144 South Dale Mabry Avenue
Tampa, Florida 33609
(813) 879-4301

Micro Store
634 South Central Expressway
Richardson, Texas 75080
(214) 231-1096

Vanguard Systems Corporation
6812 San Pedro
San Antonio, Texas 78216
(512) 828-0553

A-Vid Electronics Company
1655 East 28th Street
Long Beach, California 90806
(213) 426-5526

Electronics for Yachting
1525 South East 16th Street
Ft. Lauderdale, Florida 33316
(305) 525-3478
TWX 510-955-9484

High Technology
1020 West Wilshire Blvd.
Oklahoma City, Oklahoma 73116
(405) 842-2021

Computer Workshop
174 Ifield Road
London England
SW10 9AG
01-373-8571

Midwest Scientific Instruments

220 West Cedar • Olathe, Kansas 66061 • 913/764-3273
TWX 910 749 6403 (MSI OLAT)

☐ Before I forget...send me your catalog

Name ______________________

Address ______________________

City ______________________

State ______________ Zip __________

BY7 77

*MSI BASIC is an additional $65.00

MSI is a dealer for Southwest Technical Products • Mastercharge & BankAmericard Orders accepted

220 West Cedar • Olathe, Kansas 66061 • 913/764-3273 • TWX 910 749 6403 (MSI OLAT)

Circle 44 on inquiry card.

up to 5 mils of protective polyester coating. The company is considering the manufacture of a reader designed to read whole programs encoded on ID card magnetic stripes.■

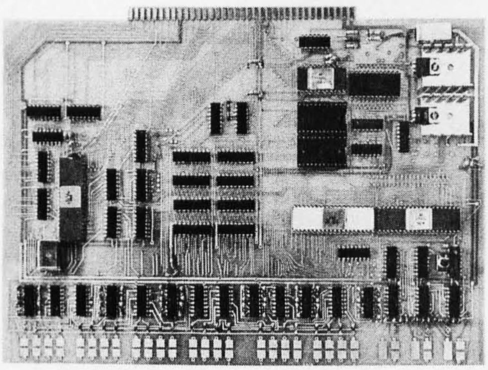
Circle 636 on inquiry card.



Attention LSI-11 People Wanting Mass Storage

Charles River Data System Inc, 235 Bear Hill Rd, Waltham MA 02154, has announced its new FD11 Floppy Disk Add-On System for PDP-11 computers as well as LSI-11 microcomputers. The system was designed to be hardware, software and media compatible with the DEC RX11 products. It includes write protect switches, unit select switches, up to four drives with one unit load on the DEC Unibus, PROM self-diagnostic programs and bootstrap loader. Track to track access time is 6 ms. Single quantity price of the drive and DEC compatible controller is $2750.■

Circle 638 on inquiry card.

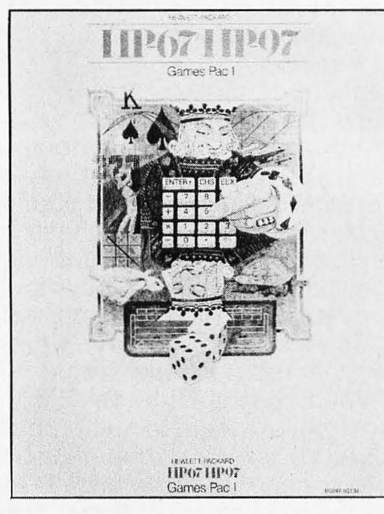


A Tutorial Training Computer

The Datac 1000 computer, controller and tutorial card is a new product available from Datac Engineering, POB 406, Southampton PA 18966. This card is available in two models, the tutorial version and the fully populated version. The tutorial version is $185 assembled and tested, with power supply; the fully populated version is $345 assembled, tested and with power supply. In either case you get a 6502 processor, an instantly usable computer with expansion capabilities, and a unique system of touch sensitive input keypads, seen along the lower edge of the board depicted in this photo. A documentation package accompanies each processor board.■

Circle 637 on inquiry card.

Portable Games



For the benefit of the steadily growing ranks of pocket calculator game enthusiasts, Hewlett-Packard is offering the Games Pac I. Intended for use with the company's HP-67 and HP-97 programmable calculators, the package contains an owner's handbook (shown) and 20 prerecorded program cards. Games include Space War, Golf, Slot Machine, Dice, and a Bowling Scorekeeper. The price is $35. Contact Hewlett-Packard, 1501 Page Mill Rd, Palo Alto CA 94304.■

Circle 648 on inquiry card.

A New Version of the PCM-12 Microcomputer

Pacific Cyber/Metrix Inc, 180 Thorup Ln, POB 215, San Ramon CA 94583, has announced a new version of their PCM-12 microcomputer called the PCM-12A. The new unit is a 12 bit machine based on the Intersil IM6100 microprocessor, and is designed to be compatible with PDP-8

software. The new model includes the following features:

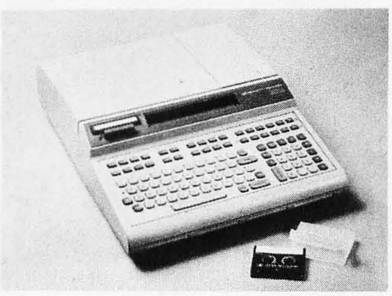
- Memory expandible to 32 K bytes of programmable memory.
- Crystal controlled data rate generator to service peripherals.
- Front panel controlled bootstrap loading.
- Can be expanded to run with Digital Equipment's OS-8 operating system.

The unit is available either assembled or in kit form. The kit price of $799 includes the control panel, 1 K byte of static programmable memory, cabinet and power supply.■

Circle 639 on inquiry card.

Hewlett-Packard's Personal Computer. . .

The HP9831A desk top computer is a personal computing entry of Hewlett-Packard, which comes in at the high end of the present day spectrum of price, $7200. This machine speaks BASIC with an 8 K byte source program string area expandable to 32 K maximum in 8 K byte increments. It uses an on board tape drive with 90 ips (229 cm/sec) search and rewind speed, 22 ips (56 cm/sec) data transfer speed. The tape cartridge can hold approximately 250,000 bytes and is available to BASIC applications programs as a resource. The average access time is quoted as six seconds. Auxilliary ROM cartridges are also available for extension of its capabilities. The main display peripheral is a 32 character LED alphanumeric device, and the keyboard is used as the primary interactive inputs. Options include printers, floppy disks and other goodies. For inquiries contact Hewlett-Packard at 1501 Page Mill Rd, Palo Alto CA 94304.■



Circle 640 on inquiry card.

Minidisk and Verbatim™ – New Products from ITC

Information Terminals Corporation, 323 Soquel Way, Sunnyvale CA 94086, announces a new proprietary ferric oxide binder system called Verbatim™. This new material will form the storage medium for the company's floppy disks, magnetic cards, computer cartridges, digital cassettes, mini data cassettes and the MD 525 minidisk. The new binder is said to be more resilient and less abrasive than previous binders.

The minidisk is approximately half the size of the standard floppy disk and provides one third the storage capacity. Priced at $5.95 per disk, the minidisk is intended for use in the small systems market as well as in power typing and editing systems. When you use high speed digital media, this type of product will prove most useful.■

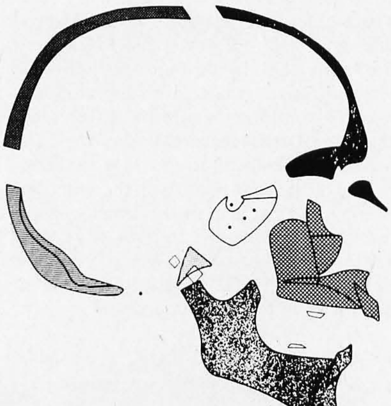


Circle 641 on inquiry card.

Bite Analysis, Anyone?

The latest issue of Complot's *Application Newsletter* describes a new technique being used at the University of Michigan's School of Dentistry. Researchers are using computer graphics

Circle 51 on inquiry card.



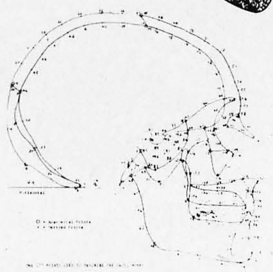


Figure 2. The coordinate points comprising our model of the lateral craniofacila morphology of the skull.

in conjunction with a computer to study how changes in the number of teeth through time can affect the shape of the skull and face.

An X-ray of the skull's profile is first digitized into 177 key coordinate points to facilitate plotting. A typical plot is then produced in 30 seconds, which can be overplotted to show deviations of the skull and jaw due to age, corrective surgery or other factors. Researcher Dr Geoffrey Walker looks upon the technique as a major tool which can improve the quality of corrective facial surgery and other dental techniques. For further information contact Complot Application News, One Houston Sq (at 8500 Cameron Rd), Austin TX 78753.■

Circle 642 on inquiry card.

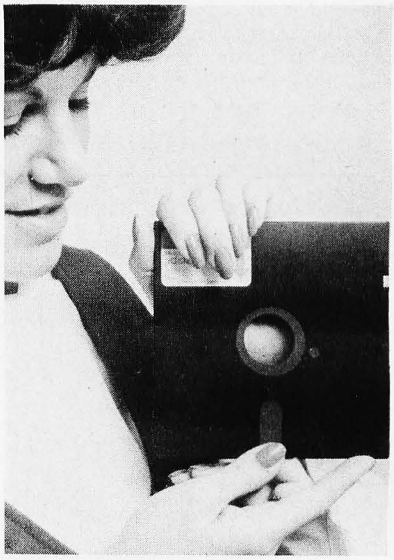
A New Economical 6800 Software Package

Users of 6800 based microcomputers will be interested in Inpro Micro Systems' new software package called MIKADOS, a combination debugging program, assembler and operating system which resides in 2.5 K bytes of programmable memory. The assembler generates object code on the same line as the user's mnemonic to provide an immediate program listing. The price, which includes user manual and hexadecimal object code listing, is $12.95. Contact Inpro Micro Systems, POB 7776, Van Nuys CA 91409.■

Circle 643 on inquiry card.

With a Small Floppy Disk, Where Does One Get Media?

Media (for those new to the term) is a generic name for tapes, floppy disks, hard surface disks, etc, which can store data. International Terminals Corporation has recently introduced a new line of floppy disk media for use with the new drives such as the Shugart Mini-Floppy. The price of the ITC MD 525 flexible disk cartridge is $5.25 in single

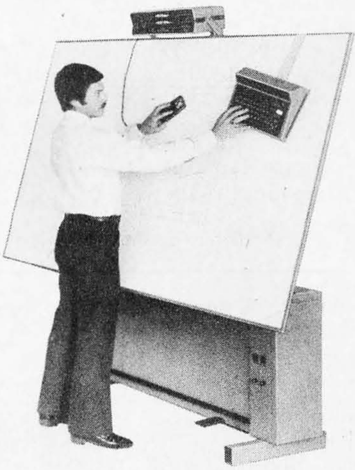


---

quantities, and the press release says delivery is from distributor stocks. The company is located at 323 Soquel Way, Sunnyvale CA 94086.■

Circle 644 on inquiry card.

Not for the Casual Amateur, But . . .

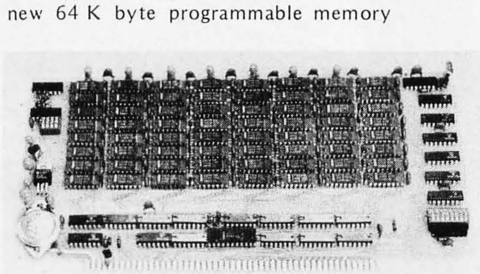


This "Traverscan" absolute free cursor digitizer from the H Dell Foster Company, POB 32581, San Antonio TX 78216, is an interesting idea which experimenters might find challenging to emulate. This is a real time graphic digitizer which measures, counts and displays the X and Y coordinate values of points, with six digits per axis read directly in English or metric units. Numerous operational features include electronic scaling, 2 axis scalable integrator (or digital planimeter) for automatic area computation, and of course, output interfaces through RS-232C serial lines so that the data can be captured by a computer. One uses a graphic digitizer to get information from drawings into a computer by tracing lines and tracing around regions. The complexity of this particular device strongly suggests it has some microprocessor intelligence built into it locally. H Dell Foster Company is a subsidiary of the Keuffel and Esser Company.■

Circle 649 on inquiry card.

Improving Your Memory

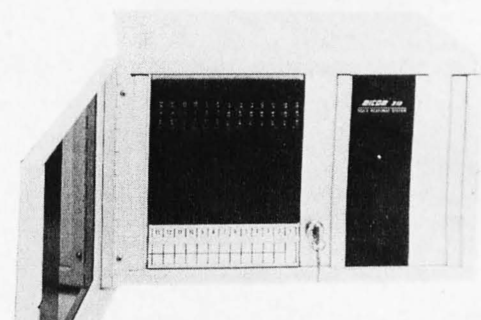
If you want to experiment with APL or other sophisticated data constructs requiring extensive memory space, (see the editorial in the November 1976 BYTE), you may be interested in the new 64 K byte programmable memory



board recently announced by Extensys Corporation, 592 Weddell Dr, Suite 3, Sunnydale CA 94086. The 5 by 10 inch (12.7 by 25.4 cm) board has a hardware provision for bank switching to add over one million bytes. This Altair compatible board also allows memory address to be set in 8 K byte increments and provides hardware protection in 16 K byte increments. Power supply voltages required are 12 V 300 mA, 5 V 750 mA, and −5 V 1 mA (Power supplies are not included). Memory overlap protection is provided to prevent conflict with existing memories. The price of the 64 K byte board is $1,495. A 32 K byte board is available for $895, and a 48 K byte board for $1,195.■

Circle 650 on inquiry card.

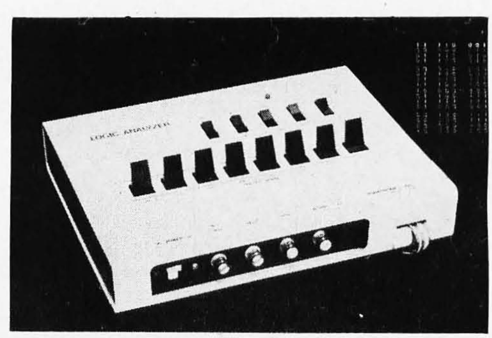
Voice Synthesizers, Anyone?



MICOM Systems, 9551 Irondale Av, Chatsworth CA 91311, is manufacturing a Voice Response System intended for minicomputers, which uses a microprocessor as a control element. The press release describes a system which can accept Touch-Tone encoded inputs and give digitally generated synthetic voice outputs. This system, which is definitely oriented toward commercial systems markets, has prices beginning at $500.■

Circle 651 on inquiry card.

A Decimal Order of Magnitude Difference in Price



We received this picture of a new Model 100A Logic Analyzer, produced by Paratronics Inc, 150 Tait Av, Los Gatos CA 95030, and seen in some detail in *Popular Electronics*, February 1977. This unit is available in a complete kit for $198.50, along with a 100 page assembly and applications manual.

According to its makers it compares quite favorably with commercial laboratory equipment products costing $2750 (and up). The output display is on an ordinary oscilloscope, with blanking, and consists of a truth table of the last 16 states of a 1 byte word of data. Various other useful features make this an interesting test instrument which may prove useful to many of our readers.■

Circle 652 on inquiry card.

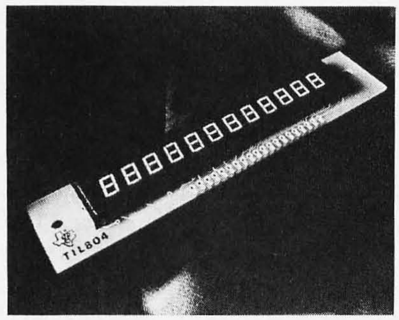
Two New DaJen Cassette Interfaces

DaJen Electronics, 7214 Springleaf Ct, Citrus Heights CA 95610, has announced two new cassette interface boards compatible with the Altair bus. The first board is a universal cassette interface called the UCRI, available for $175 assembled or $135 in kit form. Data transmission rate is user selectable from 520 to 41,000 bps. The unit can be optionally equipped with two reed relays allowing the unit to independently control two cassette recorders. The second interface board, the CRI-B, has a maximum data transmission rate of 6000 bps and is available for $165 assembled or $120 as a kit.■

Circle 571 on inquiry card.

Texas Instruments Introduces 12 Digit VLED Display Board

Thinking about your own custom decimal displays? Texas Instruments has announced a multidigit visual light emitting diode (VLED) display stick with 12 digits on a single board, called the TIL804. The characters are 7 segment red VLEDs, 0.27 inches (0.66 cm) high; typical brightness is 500 microcandelas at 20 mA. The display stick features right hand decimals at each digit, continuous uniform brightness of segments within each digit, and a wide viewing angle for distances up to 15 feet. It is presently available in a common cathode configuration to facilitate multiplexing. Intended commercial applications include Citizens' Band radios, scanners, digital instrumentation, electronic games, medical electronics, test and measurement equipment and desk top calculators; these would make



excellent numeric displays for the experimenter as well.

The price of the board is $14.65 in quantities up to 100 and $11.65 each in quantities above 100 to 999. Write Texas Instruments, POB 5012, Dallas TX 75222.■

Circle 659 on inquiry card.

A Source of Neat Ideas

Persons interested in discovering the functional characteristics of a well thought out editing terminal design would do well to look at the new "2645A Display Station User's Manual," publication number 02645-90001, put out by Hewlett-Packard Data Terminals Division. (Inquiries should be addressed to Inquiries Manager, Hewlett-Packard Company, 1501 Page Mill Rd, Palo Alto CA 94304.) This pamphlet of approximately 40 pages in length is the user's manual for this microprocessor controlled data terminal with editing capabilities, which make it extremely adaptable to numerous practical tasks.■

Circle 658 on inquiry card.

A New Tutorial Kit from Motorola

Motorola has announced a new 8 bit 6800 based microprocessor kit for tutorial purposes called the Educator II. The unit contains a 128 byte

programmable memory, a 512 byte read only memory and a TTL clock circuit.

An executive program residing in



the read only memory contains routines for servicing interrupts, program transfer to and from cassettes, tape searching programs, and a program to test the finished kit. The executive program uses 14 bytes of the programmable memory as a scratchpad memory, leaving 114 bytes for user programs. An optional 128 bytes of programmable memory can be added later.

The Educator II comes with an aluminum case for $169.95, including assembly and operating manuals. The unit is available from Motorola distributors. For further information, distributors and computer stores should contact Motorola HEP/MRO National Sales Manager, 705 W 22nd St, Tempe AZ 85282.■

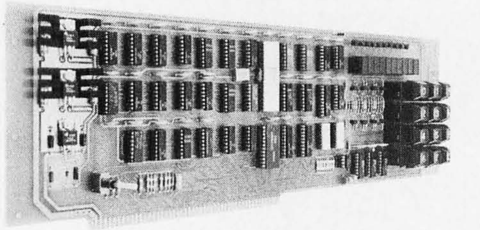
Circle 657 on inquiry card.

Timesharing Software Listing Available

Gregory Research Associates, 1900 Greymont St, Philadelphia PA 19116, are making available a directory entitled *Remotely Accessible Conversational Programs and Data Bases.* The price is $28, which includes three bimonthly updates. The listing is a guide to thousands of programs for timeshare users in the areas of business, engineering and science. The programs listed are written in a conversational style which requires no special knowledge of computer languages. Users pay only for those programs used based on rates listed in the directory. Programs are received via standard telephone lines; some require terminal data rates of 120 characters per second.■

Circle 653 on inquiry card.

A Processor Board with Expanded Features



Morrow's Micro-Stuff, POB 6194, Albany CA 94706, announces an Altair bus compatible plug-in processor board designed to expand the capabilities of existing 8080a microcomputer systems. Available for $325 assembled or $250 as a kit, the board offers two special features. The first enables the user to step through a program at a rate variable from 1 to 65,000 steps per minute. The second feature prevents the 8080a processor from shutting off after a HALT instruction. While in this state, registers, memory and IO locations can be examined and altered. A built-in 12 pad keyboard and 10 digit readout are included on the board.■

Circle 654 on inquiry card.

A New 6800 Microcomputer Evaluation Kit

A new microcomputer evaluation kit for M6800 systems is now available from the Integrated Circuit Division of Motorola Inc. This kit will prove quite useful to those individuals with a minimal budget and a keen interest in computers.

The MEK6800D2 kit, when assembled, is a fully functional microcomputer system based on the MC6800 microprocessing unit and its family of asso-



Circle 49 on inquiry card.

ciated memory and IO devices. It is made up of two basic units, a microcomputer module, 9.75 by 8.3 inches (24.8 by 21 cm), and a keyboard and display module, 10 by 6.25 inches (25.4 by 15.9 cm). The keyboard and display module also contains audio cassette interface circuitry for the Kansas City Standard, 300 bps redundant phase encoding.

The display consists of six 7 segment LED readouts that display four address digits and two data digits in hexadecimal format. The keyboard is hexadecimal with eight additional command keys.

Also included in the kit's three-ring binder is an assembly manual that covers testing, schematics, "JBUG" monitor program listing and parts listing. An M6800 Programming Reference Manual and the M6800 Microcomputer System Design Data book are also part of this package. The kit may be used "as is" or expanded to a full 64 K system through addition of buffers (and minor modifications of on board memory decoding).

Cost of the kit is $235, and it is available from Motorola Distributors. The 5 V, 2 A power supply and cassette recorder required by this device are not included, and can be purchased locally.■

Circle 655 on inquiry card.

A Summary Detail of the MEK6800D2

The eight command keys are:

- M Examine and change memory.
- E Escape from operation in progress.
- R Examine contents of processor registers P, X, A, B, CC and S.
- G Go to specified program and begin execution.
- P Transfer programs or data from memory to cassette tape.
- L Load memory from cassette tape.
- N Trace one instruction.
- V Set (and remove) breakpoints.

The Microcomputer Module includes the following devices:

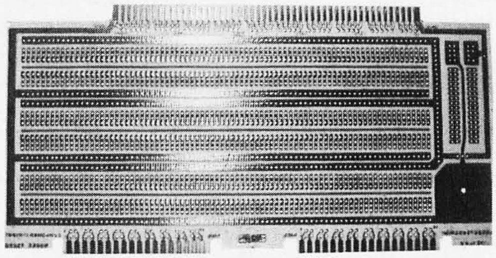
- 1 MC6800 MPU
- 1 MCM6830 read only memories with JBUG monitor (SCM44520P)
- 3 MCM6810 programmable memories (128 by 8 for 384 bytes total)
- 2 MC6820 peripheral interface adapters (PIA)
- 1 MC6850 asynchronous communications adapter (ACIA)
- 1 MC6871B clock generator

In addition, the board has been engineered to accept the following optional devices (not included with the kit):

- 2 MCM6810 programmable memories (128 by 8)
- 2 MCM68708 erasable read only memories
- 3 MC8T97 buffers
- 2 MC8T26 bidirectional buffers

Custom expansion on the Microcomputer Module by the homebrewer is simplified by a wire wrap area that will accommodate two 24 pin and twelve 16 pin sockets. The JBUG monitor read only memory can be replaced by a Mini Bug II read only memory for RS232 interface to other peripherals.

A New Prototyping Board



A new general purpose printed circuit board for the experimenter is available from World Wide Systems Corporation, 8305 Private La, Annandale VA 22003, for $24.95 postpaid. The 100 pin board is Altair bus compatible and features two gold plated 36 pin IO strips. A portion of the board has been reserved for the installation of a voltage regulator and heat sink; the power bus, ground plane and pins are labelled. Board dimensions are 5.125 by 10 inches (13.02 by 25.4 cm). The board can be used with point to point solder connections, wire wrap or a combination of the two. With this tool the experimenter can try his hand at designing custom peripheral interfaces.■

Circle 656 on inquiry card.

Christopher D Glaeser
3904 NW 15th St
Gainesville FL 32605

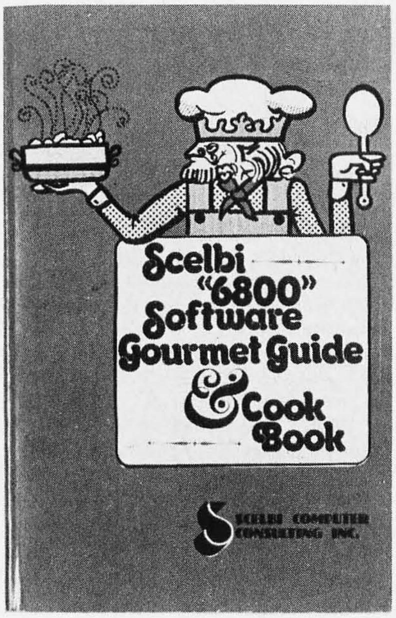
Programming Quickies

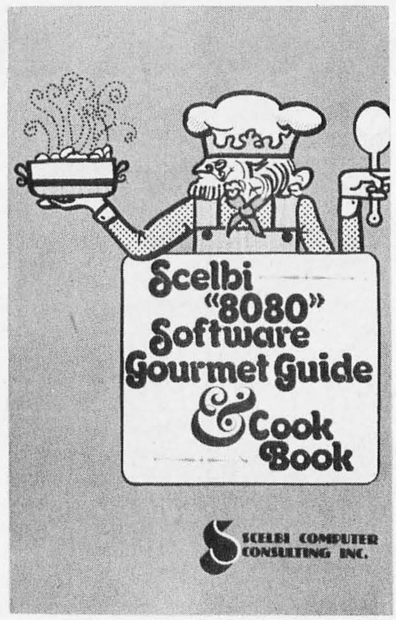
Novel 8 Bit Multiplication

Here is a 1 byte multiplication routine that was submitted by Christopher D Glaeser for use on an Intel 8080 along with his idea on how to save memory space and execution time. Working on the premise that one of the main problems with many multiplication routines is that they must call another routine to perform a double byte shift, he uses the DAD H command of the 8080 to surmount this difficulty. The DAD H command performs the double byte shift by adding the HL register pair to itself. This saves having to call another routine to perform this function. The operands are loaded into the C and D registers prior to calling the subroutine. The resulting answer is placed in the BC register pair. All of the numbers in this listing are in octal.

Address	Op	Operand	Label	Mnemonic	Commentary
006000	325			PUSH D	save original values of registers on stack;
006001	345			PUSH H	
006002	132			MOV E,D	E:=first operand;
006003	026	000		MVI D,000	initialize registers;
006005	152			MOV L,D	
006006	142			MOV H,D	
006007	006	010		MVI B,010	
006011	171			MOV A,C	A:=second operand;
006012	037		LOOP	RAR	A:=A/2;
006013	322	017 006		JNC SKIP	if CY:=0 go to SKIP;
006016	031			DAD D	else HL:=HL+DE;
006017	353		SKIP	XCHG	exchange HL with DE;
006020	051			DAD H	HL:=HL*2;
006021	353			XCHG	exchange HL with DE;
006022	005			DCR B	B:=B−1;
006023	302	012 006		JNZ LOOP	if CY:=0 go to LOOP;
006026	104			MOV B,H	else B:=H; [high order 8 bits of answer]
006027	115			MOV C,L	C:=L; [low order 8 bits of answer]
006030	341			POP H	restore original value of registers;
006031	321			POP D	
006032	311			RET	return to calling program;■

Gourmet Reading from BITS

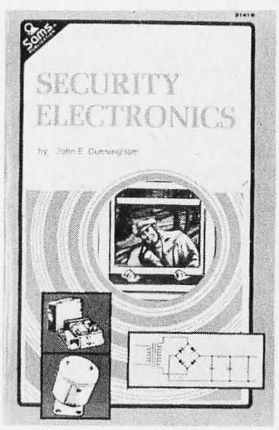


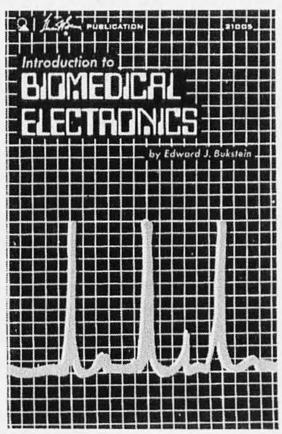


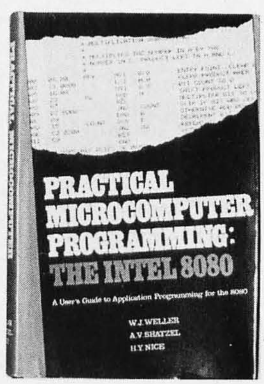
____**Scelbi "6800" Software Gourmet Guide & Cookbook**

____**Scelbi "8080" Software Gourmet Guide & Cookbook**, both by Robert Findley. Have you tried cooking up a program lately on your 6800 or 8080 processor? Have you needed a dash of ideas on how to add spice to a program? Then the **Scelbi "6800" Software Gourmet Guide & Cookbook** and the **Scelbi "8080" Software Gourmet Guide & Cookbook** may prove to be quite useful additions to your library.

Both books contain a wealth of information on the 6800 and 8080 instruction sets, plus general programming techniques applied to the 8080 and 6800, conversion routines, floating point routines, decimal arithmetic routines, and much more. Order yours today and get a taste of what your processor can do. Both books are priced at $9.95 each.







____**Security Electronics** by John E Cunningham. To catch a thief, apply liberal doses of ingenuity and a modicum of cleverness. Find out what's been tried in conventional alarm systems before you go off computerizing your home security system, though. New 2nd edition $5.95

____**Introduction to Biomedical Electronics** by Edward J Bukstein. What's been done in robot doctors? Nothing so far. But in terms of electronic aids to physicians and practices of health researchers, consult this background review of the field of biomedical electronics $5.95

____**Practical Microcomputer Programming: The Intel 8080** by W J Weller, A V Shatzel, and H Y Nice. Here is a comprehensive source of programming information for the present or prospective user of the 8080 microcomputer, an architecture which appears in the MITS Altair, 8800, Processor Technology SOL, IMSAI 8080, Polymorphics POLY-88, and other popular microcomputer system products.

After several preliminary chapters, the authors get down to practical details with topics such as moving data, binary arithmetic operations, multiplication and division, use of the stack pointer, subroutines, arrays and tables, conversions, decimal arithmetic, various IO options, real time clocks and interrupt driven processes, and debugging techniques. Most examples are given in symbolic assembly form, with occasional listings of assembled code using a Computer Automation software development system.

This 306 page hardcover book is well worth its $21.95 price and should be in every 8080 or Z-80 user's library.

Send to:

BITS, Inc
70 Main St
Peterborough NH 03458

Total for all books checked $ ________
Postage, 50 cents per book
for ________ **books** $ ________
Grand Total $ ________

Check payment method:

___My check is enclosed
___Bill my MC No. ____________________ Exp. date ________
___Bill my BAC No. ____________________ Exp. date ________

Name ____________________

Address ____________________

City ____________ State ______ Zip ______

Signature ____________________

You may photocopy this page if you wish to leave your BYTE intact. In unusual cases, processing may exceed 30 days

Continued from page 11

sets a very high goal for us. When your latest program crashes, a poster like this on the wall could give you the inspiration to hit the restart button and try again.

I find that I can justify my activities in small computing a lot easier when I can point to an illustration like this than through the idea of the "ultimate toy." One picture is indeed worth a thousand BYTES.

Robert J Retelle
2005 Whittaker Rd
Ypsilanti MI 49197

WHY NOT 8 TRACKS?

I am curious to know why 8 track tapes are not used for memories instead of cassettes. I would think that by using one channel for a clock track you could have memory locations via a clock track counter. Also by moving from track to track, access time would be saved, would it not? Please elaborate.

Kurt Kramer
8634 Lake Isle Dr
Tampa FL 33617

Sounds like a good idea. I once saw a surplus automated cartridge turret mechanism which had been used by a radio station. This mechanism, which its owner intended to refurbish, would be used for an on line storage system of rather large capacity based on the wide tape cartridges similar to an 8 track stereo cartridge. Maybe a reader will pick up on this and try it out, reporting the results as an article.

RETROSPECTIVE: BYTE'S PORNOGRAPHIC PHOTOS MARCH ON

Never thought that BYTE would go in for pornography. But without the benefit of a G-string, you show on page 54 of the March 1977 issue a bare soldering iron. Soldering irons without G can certainly corrupt innocent computers. Since your local Sears or Radio Shack do not generally bother to stock properly grounded ones, beginners should be warned.

Sholom Kass
567 Baden Av
San Francisco CA 94080

The G-string mentioned in Sholom Kass's letter is the ground wire of a 3 wire 110 V interface for power. It is standard industrial practice to use such grounding, especially where delicate parts such as microprocessors, and other LSI or CMOS gates are being used. The static charge which a normal human being can build up in a dry room with rug and rubber soled shoes can drive a spark across a considerable gap, and is measured in many tens of thousands of volts.

SOME NEWS FROM DEUTSCHLAND

My involvement with micros began when one of your associates, Dan Fylstra, gave a talk at IFIP/IFAC in Paris last summer, in which he managed to mention the Motorola 6800, the MOS Tech 6502, and also gave an example of a program in PL/M. On coming back to Darmstadt, I found two groups of people at our facility who were interested, nearly all professional computer engineers. One of them, Asbjørn Smitt, had already worked on a 6800-based system which we use for simulating satellite telemetry data; another, Helmuth Werthmann, had his own business and had already built a 6800-based system. So our choice as a club was somewhat inevitable. Asbjørn had bought the EVK 100 and the SWTPC AC-30 cassette controller, and we very nearly went EMI, but then the prices sort of doubled, so we went with OSI instead after our chairman, Graham

Else (whose main interest is biological monitoring), had convinced everyone we needed something which could be started off cheaply. (In the end Graham is at the Rolls-Royce end of the market, with a Digital Group Z-80.) I had bought a SCAMP, which to my astonishment worked first time despite some startup trouble which we attributed to the rather bulky socket for the CPU, and which eventually I hope to work up into a controller for the Selectric I'm typing this on. Our club is now about 40 strong, with considerable purchasing leverage since there are believed to be about 30 people actively buying components (25 OSI superboards, 16 Selectrics, which was all I could get in my Volvo the last time I went to England, and if you know anything about the quaint European customs, avoid Belgium), and of course our own BYTE archive. We are also in contact with John Barnes, a founder and member of the British Amateur Computer Constructors' Association, and having heard of the success he has had in using standard 8 track cassettes with short endless loops and saturation recording, I have ordered a few for experiment. Graham and Helmuth in the meantime are working on Philips cassettes.

Things would probably have started more slowly had it not been for the impact of BYTE and the fact that Jack Davies, the guy running PACS, lives in Darmstadt with a perpetual fount of goodies which are available for inspection and (usually) instant purchase. The only complaint we have about OSI is that they have never dished out the listing of the Superbug monitor; so this has now been decompiled using the Thomson Lister (running on the first of our club machines, built by Bob Dees of ICL), and another program running on our 4/72. (ICL, not Amdahl!) I guess our next move is to find out if Tom Pittman has an OSI version of Tiny BASIC yet.

I suppose the next move would be for us to get some sort of European convention going. It would be nice if I could appeal through your pages for any hobbyist clubs in Europe who would be interested in attending such a thing, or indeed if anyone is prepared to offer a site which is fairly central. There have been two attempts to get Europe-wide hobby clubs formed, but in both cases the impulse has come from the hobby industry rather than the hobbyists, who tend to view (other people's) commercial bias with suspicion. If we forget the launches which are scheduled for this year ESA can probably be persuaded to host it if we hear from enough people, but obviously we are equally curious if anyone has a 72 character graphics cum VDU interface for 625 line TV receivers, or a meta-assembler capable of compiling code for more than one micro, and so forth. Incidentally, why use OMEGA when there are satellite systems, like LORAN-c? *[Ralph Burhans is busy at work on a LORAN-c interface . . .* CH*]*

Good luck to BYTE, may it grow ever heavier, and perhaps sprout a Euro-

pean edition with the addresses of outfits here.

Ian McNicol
Waldstrasse 18
6101 Lichtenberg GERMANY

IMPRESSED, BUT

I was impressed with "The Built-In Logic Tester," Kurt Christner's article in the January issue, page 82. He had an idea for a super logic probe, considering the cost.

But what about those amateurs among us who have need of a CMOS probe rather than TTL? I would like to challenge Kurt to come up with an equally fine project for us nontechnically inclined who could really use a good quality, low cost CMOS probe.

By the way, I'm brand new with the January issue and I like what I see. It's a great magazine.

Tom Kryst
212 Princeton
Alpena MI 49707

COMPUTER POSSIBILITIES IN SECONDARY SCHOOLS

Attached is something that appeared recently in *Science News* (see page 168, March 12 1977 issue) which is a semi-popular weekly publication. What I found particularly interesting was the section concerning this year's winners of the national Science Talent Search. Of the ten winners, notice that four of them used computers in some capacity. Thinking back on my science fair entries of 20 years ago, such a thing borders on the inconceivable. Until very recently computers and computer time were so expensive that the economics of the situation precluded any but the work which had a very high probability of research or financial return. Therefore such investigations as these by high school students were impossible to justify on the basis of what they might produce.

Perhaps one of the things that those of us involved with the microcomputer and personal computing revolution tend to forget is that to most of the academic and scientific world at large, computers are simply tools to an end in whatever discipline is being pursued. Just think of what the possible benefits are should virtually limitless, although perhaps slow, computing time become available to any interested individual, such as these high school students. A local school could put together quite a respectable system of moderate capability for, say, 10 K bucks, and then allow appropriate students to pursue their interests generally unhampered by the restraints of financial return. The leap ahead in science education (and possibly science related areas of liberal arts) is truly mind boggling!

Robert R Wier
POB 9209
College Station TX 77840

I HATE BASIC, A POISON KEY LETTER

I wish to express my appreciation for your publication. For me, it has helped open the door to a new and fascinating world.

I wish to pose a general question. Why is it that, with all too few exceptions, the thrust of what might be called "microcomputer applications for the nonprofessional" seems to be lost in an endless loop called "games written in BASIC"? And as a corollary, one might well ask the question, "Of what redeeming social value is the latest listing of Star Trek?".

The Pong-Trek-Toe mentality seems to be ubiquitous; lift any magazine cover and underneath, in the table of contents, will be found an article dealing with games written in BASIC.

This is most assuredly not an attack upon BYTE magazine. It most assuredly is an attack upon the paucity of imagination manifested by microcomputer owners.

For every ten serious articles appearing in BYTE which are devoted to the embarrassment of riches in hardware being placed within reach, there is perhaps one serious and informative article devoted to software concepts. The term "concepts" should be emphasized; an assembly language listing for a specific microprocessor is not the expression of a concept.

The lack of interest in software concepts coupled with the overweening interest in amusing trivia does not speak well for the present state of the amateur group, from which emanates a great talk about "being in the vanguard of a revolution." Most of this talk is vainglorious nonsense.

At the present time, the only groups doing things worthy of the adjective "revolutionary" are the microprocessor manufacturers and the microcomputer system designers. For the purchaser of such riches to sit around playing Star Trek in BASIC borders on the perverse; to ascribe to such apathetic behavior the word "revolutionary" is a caricature.

It seems that the amateur group has been handed BASIC on a silver platter and is content to view it as *the* programming language. This is unimaginative at best.

The reason for being of microcomputer BASIC is simply to provide a workable language so that people will buy microcomputers. Insofar as I know, no one has ever claimed that BASIC uses the resources of a system organized in 8 bit words to its maximum benefit; BASIC has appeared on the microcomputer scene simply by default. It works and it is easy to learn and use. It sells microcomputers.

If Star Trek is the desired goal of our collective amateur ambition, then BASIC is sufficient; we need look no further.

However, if we wish to exploit the limited resources of the microcomputer in specific ways, looking toward specific

nontrivial goals, then new programming languages should be written. There surely exists a diversity of ideas as to what constitutes good microcomputer programming languages. Attempts should be made to implement these ideas. We cannot sit back and wait for commercial organizations to do all the development.

Most of all, we need software concepts. Memory management concepts. Data manipulation concepts. Algorithms for mathematical functions.

What is a heap? How does it work? What does its presence imply about the philosophy of a language which uses it?

If, for every page devoted to Star Trek listings and other compatible endeavors, BYTE published a page devoted to software concepts, then we would all soon be the richer.

However, BYTE magazine cannot prosper by serving up beefsteak if the readership is clamoring for canned pork and beans.

From time to time, in various magazines such as BYTE, there appear wild-eyed letters to the editor accusing the programming profession of clannishness, a lack of willingness to share ideas, secretiveness and a general depravity. Such letters always imply that we, the amateurs, are pure of heart and clean of hand. Is this true, or is it rather that we are so numbed by sitting around playing with BASIC (implemented by professional programmers) that we really just don't have much of any deep worth to share with each other, and seek to hide the fact by freely sharing what little we do have. Perhaps purity of heart and cleanliness of hand grow as the inverse of the amount of real knowledge which is available for sharing.

Why the great emphasis upon BASIC? One would think that most microcomputers are hard wired for it. Have we no alternative ideas to explore? Is the fixation upon BASIC a result of its inherent superiority as a programming language for microcomputers, or is it due to a lack of knowledge concerning possible alternatives? The set of "possible alternatives" is not the universe of FORTRAN. We do not need FORTRAN. We do not need miniaturized versions of languages originally designed for implementation upon much more sophisticated hardware. We need languages based upon the material at hand: 8 bit word length microprocessors with relatively primitive instruction sets.

While the Micro-Soft venture into APL represents a noble undertaking, it nevertheless embodies the faulty reasoning that, "If a language implemented on big computers is a good thing, then its implementation on a microcomputer must be equally good." In a 64 K system, what percentage of memory is required to support APL? What is left over for the user?

The microcomputer needs software which is conceived of in terms of the microprocessor. We, the amateurs, need to get up off our hind legs and do some thinking about language concepts.

In the future, is the personal microcomputer to be regarded as a tool for learning, or is it to be a toy in front of which the owner sits, glassy eyed, playing the latest game written (by someone else) in BASIC?

Jack Cliff
34-57 73rd St
Jackson Heights NY 11372

What Are Riches For, If Not to Enjoy and Prosper By?

Regarding the BYTE magazine content, I seem to hear comments from all directions concerning too much of this, too much of that, not enough of this. Some say too much hardware, some say too much software – so the conclusion can only be that the balance is a good approximation of what is needed.

No one ever seriously should think of FORTRAN or BASIC as the "be-all and end-all of high level languages." We do have demonstrated interest and familiarity on the part of users with these languages due to their widespread past usage. But, if you want to write an interpreter or compiler, starting with an existing language is a shortcut (quite independent of marketing considerations) that bypasses the need to generate an unambiguous grammar and precise semantics of a new language from scratch.

Of course, the ultimate is to use an already established language for which you have an interpreter or compiler. The fact that I can use my BASIC for an occasional recreation or game by no means prohibits my use of that language for such "unnatural" things as systems programming. I know of at least one 8080 assembler written in BASIC, running on an Altair 8800 floppy disk system at a local college.

Another point worth closing on: Most professionals are really amateurs who have cleverly arranged things so that they get paid for what they like to do. The spirit of involvement with the activity, doing it "right" by some standard, is what counts . . . CH

1001 TALES OF
ARABIAN BYTES?

Thank you for a very interesting magazine. I am a recent subscriber having just received my first two issues, January and February 1977, forwarded to me from Texas. I am unwilling to wait for boat mail. When subscriptions are available by air mail to Saudi Arabia I will enter a subscription immediately regardless of the mail rates.

I have a small system running and am interested in all articles about Altair bus oriented products and 8080 software.

John R Fogle Jr
Box 403, c/o Aramco
Abqaiq SAUDI ARABIA

See the subscription card for foreign air mail rates. ■

Technical Forum

APL Character Sets

The letter from Roderick Montgomery concerning the availability of a special character set for APL started some thoughts on the subject.

Lacking the availability of a PROM type character generator, there are other ways to do the job. The methods that come to mind are building up a generator out of standard bipolar ROMs, or using ordinary programmable memory parts which would be loaded by the computer at APL initialization time. The latter idea is especially appealing to me, since it would allow special characters to be carried into the machine as part of the program. Although this would require some programming, it would not use up main memory space because it would be read directly into the character generator.

Charles J Billwiller
2313B Sierra Madre Ct
Rancho Cordova CA 95670

The use of reprogrammable character generators for television displays is starting to occur in products. One of the first products seen in advertising with this feature is the ECD Corporation's computer product, which has just such a user memory definable character generator. This technique has also been used by several video games manufacturers to achieve high resolution regions for graphics on a TV raster.■

Is This a Valid Hot Board Placement Procedure?

A reader has suggested the following procedure to allow one to plug in and remove a circuit board while system power is maintained. We'd appreciate comments from some of the hardware oriented readers in the audience about the safety of this algorithm:

Removal of board:

0. Halt processor to avoid software carnage.
1. Connect jumpers from main supplies to board terminals to retain power during removal.
2. Remove board with care about alignment.
3. Remove jumpers to power down the board.

Insertion of a board:

0. Power up the board with jumpers from main supplies.
1. Insert board carefully to ensure proper alignment of edge connector.
2. Remove jumpers.
3. Restart processor.

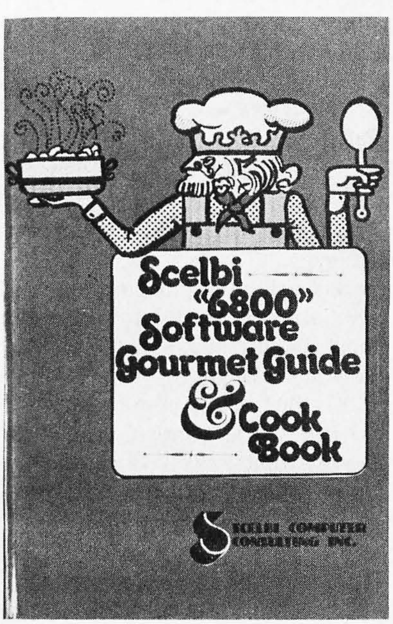
The use of a parallel supply connection to the board during removal and insertion operations guarantees that the integrated circuits will be powered up before any low impedance driving voltages reach the inputs of onboard gates. A gate can be blown if voltages are applied to its inputs prior to the supply voltages. The common ground of the supply also eliminates static voltage differences between the two boards.■

Book Reviews

Scelbi's "6800" Software Gourmet Guide & Cook Book, *by Robert Findley, Scelbi Computer Consulting Inc, Milford CT, 1976, 5½ by 8½, 226 pages. Softbound, $9.95 postpaid.*

I suppose everyone who writes assembly language programs which eventually work becomes convinced he or she is the World's Greatest Programmer. At least I do. Luckily, there are books like *Scelbi's "6800" Software Gourmet Guide & Cook Book* to remind us how big the world really is. Author Robert Findley and associates at Scelbi have collected here a number of programming tricks which will interest almost any 6800 user.

Following a description of the 6800 instruction set and some general techniques, the author discusses conversion, floating point and decimal arithmetic, input and output operations, search and sort routines. Want an edit program? Many of the pieces are there: memory clearing, transfer of a section of memory, ideas for search routines. Need to process interrupts? The IO chapter has a discussion of interrupt processing which will be useful to anyone using the Motorola MIKBUG operating system read only memory. A chapter on conversion routines contains software for ASCII to



Baudot, as well as the more usual BCD to and from binary.

For me, the best was chapter 5, "Floating Point Routines." Although (as the author notes) there are places where the code could be shortened, it is not that bad: a complete add, subtract, multiply and divide package (with conversions to ASCII, and from ASCII to floating point) in about 1.5 K bytes of relocatable code. The entire package is assembled (the only assembly in the book) with a hexadecimal listing in an appendix.

One reason the code is so short is the extensive use made of the index register. The stack, on the other hand, is not used at all (except for the automatic usage implied by each subroutine linkage). Since 6800 indexed addressing is relatively slow, do not expect the floating point package to be fast. Yet, it's not bad: Floating point multiplication, for example, takes a little more than 800 cycles, not quite 2 ms on a SWTPC 6800 system.

One curious feature of the 4 byte floating point word format used here is that the words are stored upside down, that is, if the least significant byte of the mantissa is stored at word N, then the rest is stored at N+1 and N+2 (with the sign being the most significant bit of byte N+2). The two's complement power of two exponent is stored at N+3. This is the wrong order for the 6800 for the following reason: One (tricky) way to increment a 2 byte word is to transfer it to the index register and increment that. The author knows this trick (as shown on pages 3 to 5), but apparently does not know that the index register load instruction LDX transfers the 16 bit contents at memory location M so that the most significant half of the index register contains the contents of M, and the least significant half contains what's in M+1. Thus, the least significant byte needs to have a greater address. The division program on page 5 to 19, for instance, might be shortened and speeded up by using this trick; to do so, however, would require storing the mantissa bytes in the opposite order, and rewriting the program.

The book will be of most use to programmers who employ an assembler program, since most of the ideas are presented in symbolic source language form. (The major exception is the floating point package.) The book is well written and, considering the diversity of topics, well organized. As a source of ideas, it is inexpensive at $10.■

Jack Bryant
Mathematics Dept
Texas A & M University
College Station TX 77843

Note. . . . *Scelbi also publishes an "8080"* Software Gourmet Guide & Cook Book

---

dr. dobb's journal of $1.50

COMPUTER

Calisthenics & Orthodontia

Running Light Without Overbyte

August, 1976 | Box 310, Menlo Park CA 94025 | Volume 1, Number 7

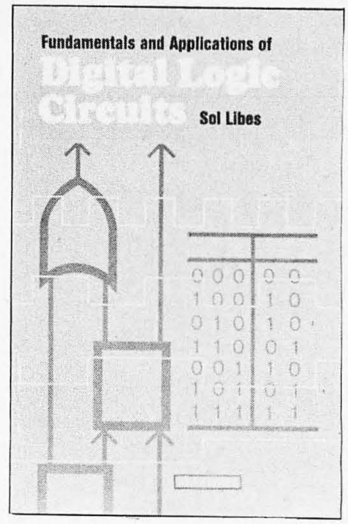
A REFERENCE JOURNAL FOR USERS OF HOME COMPUTERS

Dr Dobb's Journal of Computer Calisthenics and Orthodontia. *Published by People's Computer Company, Box 310, Menlo Park CA 94025. Single issue, $1.50; one year (10 issues), $10.*

Published by the people who brought you the Computer Faire, this toothsome periodical serves up an amalgam of filling software listings and highly spiced commentary to its monthly readership. *Dr Dobb's Journal* (Dr Dobb is disguised as editor Jim Warren) serves as a clearinghouse for new software information and the publication of such royalty-free programs as BASIC interpreters for use by readers. Because it is a nonprofit journal, *DDJ* accepts no advertising but does print new product information as a service to readers. Editor Warren has attempted to bridge the gap between computer professionals and amateurs, and does not eschew the occasional biting remark in doing so. In Robert Benchley's immortal phrase, the tooth, the whole tooth and nothing but the tooth about *Dr Dobb's Journal of Computer Calisthenics and Orthodontia* is that it can help make your computer an indentured servant. That about caps off this review . . .CM

Fundamentals and Applications of Digital Logic Circuits *by Sol Libes, Hayden Book Co, Rochelle Park NJ, 1975, 184 pages. Paperback, $6.95.*

If anyone were to ask me to pick a particular book from my computer book



library and say, "This book gave me my money's worth," I would have to say that the book would be *Fundamentals and Applications of Digital Logic Circuits* by Sol Libes. Anyone who is just starting to learn about digital electronics or computers should make it a point to study this book. Mr Libes has taken a multitude of related subjects and blended them into a text that is easy to understand and just as easy to follow. In addition to the text, the book provides review questions at the end of each chapter and problems related to the text for the reader to solve.

Rather than jumping right into digital logic, the author starts out with the principles of semiconductors to show, in detail, how they function. Then, after this short course on semiconductors in the first chapter, the second chapter familiarizes the reader with binary numbers and coding systems. This particular chapter also shows how to convert numbers from one system to another and touches on the Gray code. These first two chapters establish the foundation for a great deal of the information the reader will receive from the remainder of the book.

During the next four chapters the reader learns the basic fundamentals of logic gates, flip flops, counters and registers, and arithmetic logic circuits. Throughout the book each fundamental is illustrated and discussed in great detail. Through these same well done illustrations the reader next learns about various pulse sources and clock systems used in computers. Further, the author goes on to explain such devices as read only memories, as well as input and output equipment for computers.

Continuing with this stage-by-stage progression, the author then enlightens readers about the circuitry and theory of digital to

analog conversion. This tenth chapter also deals with multiplexing and digitally controlled analog devices. Finally, in the last two chapters of his book, Mr Libes shows the reader more of the applications aspect of digital logic circuits. These two chapters explain the different types of circuitry used in digital voltmeters, multimeters, calculators and computers (the latter being a Digital Equipment PDP-8/E).

Throughout the book the reader is taken in step-by-step fashion from the basics of transistor workings to circuit applications in working digital machines. As a result, this book is one from which the beginner will be able to learn the fundamentals and build on them afterwards.■

Michael P Reardon
17 Earl Ln
Rothsville PA 17573

BYTE's Bits

ACM Pacific 77 Conference

Small computers, from programmable handhelds through mini and micro networks, are the subject of technical papers sought for this year's ACM PACIFIC 77 conference of the Association for Computing's Pacific Region chapters. The San Francisco Bay Area meeting will be held at San Jose's LeBaron Hotel July 28 to 29 1977.

"Exploring the Small Computer" is the theme of the conference which is expected to range in coverage from personal computing through small business applications and from computer parts and peripherals through bullet proof software. Papers will be presented on new developments in software and hardware in these areas and on minilanguage processors, miniperformance predictions, microoperating systems, multimicroprocessor systems, packaging of software for sale, portable microsoftware and software engineering "in the small." Papers on trends in software and current application, on the future of minis, and on related small computer topics are also to be presented.

Peter Szego, Ampex Corporation, is General Chairman of this year's regional conference, which is jointly cosponsored by the Association's Pacific Region, Peninsula and Golden Gate Chapters. Informal symposia, workshops and invited papers are expected to be scheduled for the meeting in addition to the technical paper sessions. Special conference feature will be an evening "hobby computer" session, to be arranged by Jim Warren, editor of the home computer users magazine, *Dr Dobb's Journal of Computer Calisthenics & Orthodontia.*■

START HERE, SCAN DOWN PAGE

FRAME IDENTIFICATIONS

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32

Dr Welles' Economy Floppy Disk Drivers: Machine Readable Object Code

Last month, we published Kenneth Welles' article on the driver software for his Economy Floppy Disk. This issue, we continue that article with the machine readable bar code representation of the object code for the programs. These programs were presented as complete symbolic assemblies of the programs on pages 92-96 as "listing 1" of Dr Welles' article in June 1977 BYTE. The materials presented here are:

←

Figure 1: Bar code representation of the object code. This block of machine readable bar codes contains information in the following format in each line of bars, which we call a "frame:"

- *Sync character, binary 10010110*
- *Frame checksum, 8 bit summation of all remaining bytes in the frame, ignoring carry out of the high order bit.*
- *Relative frame indentification, 8 bit ascending integer enumerating frames in the block of bars printed here.*
- *Length of frame, 8 bit number, "n", giving the number of data bytes to follow.*
- *Data of frame, a total of "n" bytes as follows:*
  - *High order byte of 2 byte data address field.*
  - *Low order byte of 2 byte data address field.*

The bit level format of the data is as follows: Each bit is defined by a bar and its following space. If the bar width is equal to the following space width, the data is a 0 bit; if the bar width is three times the following space width, the data is a 1 bit.

And a Bar Code Bug...

Thanks to the efforts of Ken Budnick, we found one fairly serious error in the generation of the texts printed in the March and May BYTEs. Ken has implemented detailed scanning programs for several microprocessors and one antiquated second generation machine and has tested them with a bar code reader design by Fred Merkowitz. The bug, which Walter Banks has since corrected in the phototypesetter driver programs he used to create the texts, is as follows: the checksum field of the bar code frames printed in those two issues omitted the frame identification byte and the data length byte from the calculation. Thus, to verify the checksum actually generated in the object text of the Tiny Assembler and the Bar Code Contest string, only the data field should be used to calculate this sum. The present bar code text generated for Dr Welles' floppy disk driver routines calculates the checksum properly, and future examples will reflect this fix as well.■

Table 1: Confirmation copy. This table, which was created by the same computer driven typesetter which prepared the bar code copy, contains a complete listing of the object code in tabular form. This table is not a direct mapping onto the bar code frames; it was set with fixed length lines, preceded by an address value for the first byte on the line.

```
0000 E3 F5 C5 D5 7E 23 E5 F5 21 18 00 CD 45 00 F1 CD
0010 1F 00 E1 D1 C1 F1 E3 C9 0D 0A 45 52 52 4F D2 F5
0020 0E 20 CD 00 00 F1 F5 0F 0F 0F 0F CD 33 00 F1 CD
0030 33 00 C9 F5 E6 0F C6 30 FE 3A DA 3F 00 C6 07 4F
0040 CD 00 00 F1 C9 7E B7 C8 F5 E6 7F 4F CD 00 00 F1
0050 F8 23 C3 45 00
E000 F3 06 02 C5 CD 15 E0 CD 6F E0 C1 C8 05 F2 03 E0
E010 CD 00 00 01 C9 3A 65 E2 CD 2B E1 21 4B E2 AF 06
E020 10 77 23 05 C2 21 E0 36 81 2A 66 E2 22 48 E2 21
E030 5B E2 CD F3 E0 EB 22 72 E3 CD D8 E1 CD 09 E2 11
E040 36 01 21 4B E2 AF D3 F1 3A 46 E2 F6 0A 4F CD 1A
E050 E2 79 D3 F3 E6 FD D3 F3 D3 F4 7E D3 F0 23 1B 7A
E060 B3 C2 58 E0 3A 46 E2 E6 F7 D3 F3 32 46 E2 C9 CD
E070 A3 E0 C0 21 75 E3 11 5B E2 06 00 1A BE C0 13 23
E080 05 C2 7B E0 C9 F3 CD A3 E0 C8 CD A3 E1 CD B2 E1
E090 CD A3 E0 C8 CD B2 E1 CD A3 E1 CD A3 E0 C8 CD 00
E0A0 00 02 C9 3E 02 32 45 E2 CD B4 E0 C8 21 45 E2 35
E0B0 F8 C3 A8 E0 3A 4A E2 CD 2B E1 CD D8 E1 CD 09 E2
E0C0 11 1A 01 21 75 E3 3E 81 D3 F2 CD 1A E2 DB F3 DB
E0D0 F4 DB F0 77 23 1B 7A B3 C2 CF E0 21 75 E3 CD F3
E0E0 E0 3A 3A E2 32 7F E3 2A 8C E4 C3 ED E0 7C 92 C0
E0F0 7D 93 C9 01 16 01 11 00 00 7E E5 C5 AB 47 0F 0F
E100 0F 0F 4F A8 E6 F0 AA 6F 79 07 E6 1F AD 6F 78 07
E110 E6 01 AA AD 57 79 E6 0F A8 5F 79 A8 07 E6 E0 AB
E120 5F C1 E1 23 0B 78 B1 C2 F9 E0 C9 FE 08 F2 85 E1
E130 21 3A E2 BE C8 F5 E5 3E 08 BE FA 46 E1 4E 06 00
E140 09 23 3A 47 E2 77 D3 F6 E1 F1 77 4F 0F 0F 0F 32
E150 46 E2 D3 F3 DB F1 E6 04 C2 85 E1 09 23 7E 32 47
E160 E2 FE 4D F8 DB F1 E6 04 C2 84 E1 CD A3 E1 CD A3
E170 E1 CD A3 E1 CD A3 E1 CD B2 E1 DB F1 E6 01 C2 77
E180 E1 32 47 E2 C9 CD 00 00 03 C9 3E FF 32 3A E2 3C
E190 F5 CD 9C E1 F1 3C FE 08 C2 90 E1 C9 CD 2B E1 CD
E1A0 64 E1 C9 21 47 E2 34 3A 46 E2 F6 04 32 46 E2 C3
E1B0 BE E1 21 47 E2 35 3A 46 E2 E6 FB 32 46 E2 F6 10
E1C0 D3 F3 E6 EF D3 F3 CD CA E1 C9 01 FF 03 AF 0B B9
E1D0 C2 CE E1 B8 C2 CE E1 C9 3A 48 E2 FE 4D F0 21 46
E1E0 E2 7E E6 FE 77 3A 48 E2 DE 2C F2 F1 E1 7E F6 01
E1F0 77 3A 47 E2 47 3A 48 E2 B8 C8 F2 03 E2 CD B2 E1
E200 C3 F1 E1 CD A3 E1 C3 F1 E1 DB F1 E6 20 D3 F5 C0
E210 CD CA E1 CD CA E1 CD CA E1 C9 DB F1 E6 10 C2 1A
E220 E2 3A 49 E2 E6 1E 47 DB F1 E6 08 C2 27 E2 05 F8
E230 DB F1 E6 08 CA 30 E2 C3 27 E2 00
E243 00 00 00 00 00 00 00 00
E25B 81 46 49 4C 4E 41 4D 45 58 54 00 00 00 00 00 00
E26B 00
E270 00
E375 81 46 49 4C 4E 41 4D 45 58 54 00 00 00 00 00 00
E385 00
E38A 00
```

BYTE's Bits

National Student Programming Contest

The First Annual National ACM/UPR Student Programming Championship Contest took place February 2 1977 in connection with the Computer Science Conference in Atlanta GA. The contest was sponsored jointly by the Committee on Student Chapters and Memberships and Upsilon Pi Epsilon (National Computer Science Honor Society). Teams participated from: Catawba College, Chattanooga State Technical Community College, Clemson University, Georgia Institute of Technology, Georgia State University, Louisiana Technical University, Manhattan College, Michigan State University, North Carolina State University, Purdue University, Taylor University, Texas A & M University, University of Georgia at Athens, University of Missouri at Rolla, University of New Mexico at Albuquerque, and the University of Wisconsin at Platteville. The contest was conducted over a seven hour period from 5 PM to midnight on February 2. The facilities were provided by the Computer Center at Georgia Institute of Technology and the department of Information and Computer Science. Machine time was provided by Control Data Corporation on the Georgia Tech Cyber 74 computer. The teams were given four problems to solve using ANSI FORTRAN. These problems consisted of a Conversion from Roman to Arabic Numerals, A Character Manipulation for Rearranging of Names and Titles, the Determination of Amicable Numbers, and the Automatic Scoring of the Game of Bowling. The winners were determined by penalty points and the elasped time taken for each problem. Although many teams were close on several solutions, only four teams completed three of the problems.

The National Champion Team is Michigan State University, who also won their regional competition. Second place went to Purdue University, third place to the University of Missouri at Rolla, and the fourth place to Georgia Tech. Trophies and certificates were presented to the participants at an Awards Banquet held February 3 at a luncheon in the Marriot Hotel. The participating teams also received free registration to the Computer Science Conference and attended two nights of an informal social sponsored by the Committee on Student Chapters and Memberships.

Plans have been made to make this an annual event in connection with the Computer Science Conference. In 1978 it will take place on February 22 at the Plaza Hotel in Detroit MI. Regional contests will be scheduled for the fall of 1977 to qualify teams to compete in the National Contest. Teams and sponsors for these qualifying regional contests are being solicited from all regions within ACM. If you are interested in participating or holding such a contest, please contact Dr Richard Newman, Academic Computing Services, Southern Illinois University, Carbondale IL 62901, (618) 536-2323.■

Attention Educators with a Message

The National Association of Computer Applications to Learning (NAUCAL) will hold its 1977 annual convention in Dearborn MI on November 2 thru 5 1977. The convention will focus on educational computing, simulations in education, instructional materials and teaching strategies. Sessions that describe and illustrate computer applications in learning will be given special consideration.

Individuals who would like to present or who would like to suggest others who could present may write to John S Camp and Lary Smith, Conference Cochairmen, Wayne County Intermediate School District, 33500 Van Born Rd, Wayne MI 48185.■

Repairing Torn Paper Tapes

Translucent paper tapes are an inexpensive, efficient way of permanently storing information for a computer. However, after pulling the tape through a hand held paper tape reader a dozen or so times the tape tends to get pretty ragged. Sooner or later the tape will tear and have to be repaired before it can be used again. I have found that transparent Mylar movie film splicing tape is perfect for the repair job. The splicing tape is clear, strong and will not crack or yellow with age. It is also quite thin and will not interfere with the passage of the paper tape through the reader. The tape sticks very well to paper but if necessary can be removed without causing any damage. Mylar movie film splicing tape is available at any camera store.

Richard E Hetherington
637D Pendleton Av
Chicopee MA 01020■

Michigan Ave. & Boardwalk



Shelburne Hotel
P.O. Box 1138

Atlantic City, NJ 08401

(609) 344-8131

Saturday night Banquet of Filet Mignon $17.50
Over 150 Exhibits and 60 hours of seminars

Conferences being planned on; Clubs; Newsletters; Energy Conservation; Medical Applications; and more. Seminars being planned on; all aspects of Computing; Music; Basic Programming; Games; Education; Various Microprocessor Evaluations; Synthesis; S-100 Bus; and more.

Another 1st, **Personal Computing College ™** To be held at the Shelburne Hotel.

WEDNESDAY-THURSDAY-FRIDAY, AUGUST 24-25-26
3 Day Micrompuer Interfacing Workshop by TYCHON Inc.

A hands on experience for the participants where they will spend almost 50% of their time working on well documented Interfacing and Software experiments. Students deal with the microcomputer at the bus level, interfacing the computer using solderless breadboarding techniques and assembly language programs.

Presented by Jonathan A. Titus and Dr. Christopher A. Titus, Authors of the famous "BUGBOOKS."

For reservation information and complete brochure contact: TYCHON Inc. c/o Shortess-Rawson, Dept PC77, P.O. Box 2203, South Hackensack, N.J. 07606.

FRIDAY, AUGUST 26
6 Hour seminar by Dr. Adam Osborne

"Microprocessors — Where they came from and where they are going. An analysis of all products on the market today."

Dr. Adam Osborne will present a 6 hour seminar covering topics such as: Bringing order out of chaos; All microprocessors are not equal—each serves one market better than the other; Identifying those markets best suited to each microprocessor; Real sales volume anticipated for 1977; Comparisons including: 8085 vs. Z80, 8048 vs. F8, Etc.

For reservation information and complete brochure contact:

Osborne & Associates, Inc., Dept. PC77, P.O. Box 2036, Berkeley, Ca. 94702, 415—548-2805.

PC '77 Weekend admission at the door will be $10.00. Register before August 10th and SAVE 20% and AVOID WAITING IN LINE! Admission includes exhibits and seminars for both days, August 27—28th.

Please rush __________ advance registration tickets for August 27—28th @ $8.00 each.

Please rush __________ Banquet tickets for Saturday, August 28th @ $17.50 each.

Send reduced Hotel rate information __________________.

Make all checks payable to "PERSONAL COMPUTING 77" and mail to:
PC'77, Route 1, Box 242, Mays Landing, N. J. 08330.

Name ______________________________ Amount enclosed __________

Address ______________________________

City ______________ State __________ Zip __________

Clubs and Newsletters

Conducted by
Peter Travisano

Arizona Computer Society

The Arizona Computer Society meets regularly on the second Tuesday of each month at De Vry Institute, 4702 N 24th St, Phoenix AZ 85063.

A New Club in Boston

With all the computer activity in the Boston area this should be welcome news. A new group called the Boston Computer Club is forming for people with both hobbyist and industrial interests. As of now the club provides access to a timesharing PDP-8 and an Altair 8800a. For further information contact Jonathan Rotanberg, 17 Chestnut St, Boston MA 02108.

DUMPS in Delaware

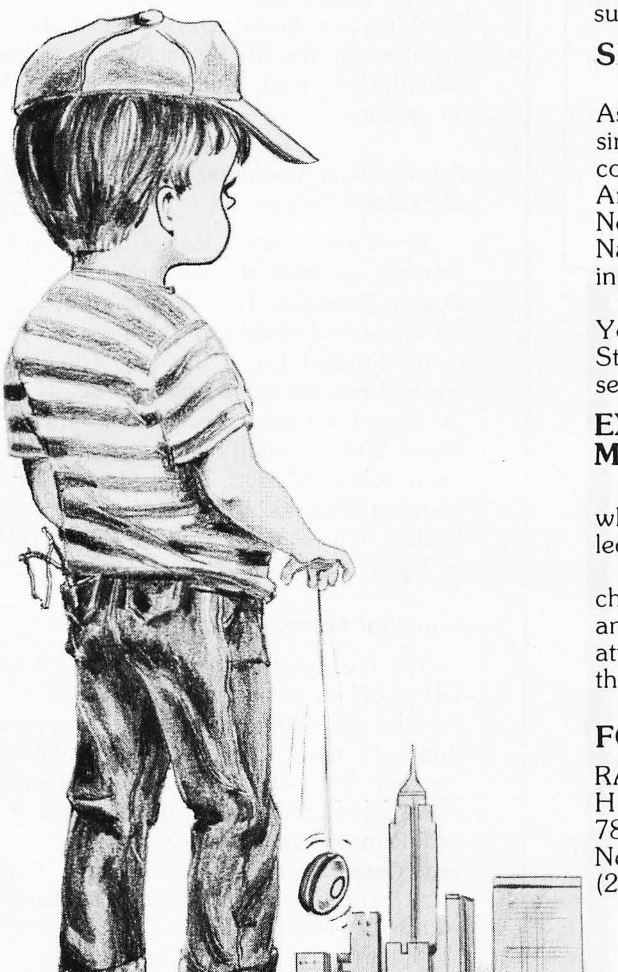
A group of hobbyists at the University of Delaware recently formed the Delaware Users of Microcomputer Systems or DUMPS. The club is informal and the membership has a wide range of interests. Meetings are held on the third Monday of the month at 7:30 at New Central School, Academy St, Newark DE 19711.

Long Island NY–Licus

The Long Island computer group is a growing concern made up primarily of high school and college students. Meetings are usually held on second Tuesdays at Commach High School South. To find out more write LICUS, POB 322, East Northport NY 11731.

State College PA

The Nittany Amateur Radio Club of State College PA sponsored a one day microcomputer seminar, an introduction to interfacing, programming and applications with an emphasis on real world situations. The



guest instructor was David G Larsen, WB4HYJ, from the Department of Chemistry, Virginia Polytechnic Institute and State University. Larsen is a member of a team that has toured the nation presenting similar seminars and has been published widely on the subject.

One significant by-product of the seminar was the formation of the Center County Computer Club. Membership can be obtained from Carl Vesper, 131 Sowes St, Apt E-10, State College PA 16801.

SR-52 Users Club

The SR-52 Users Club is a nonprofit loosely organized group of SR-52 and SR-56 users who wish to expand their knowledge through information exchange. Activity centers around a monthly newsletter, *52 NOTES*, published by Richard C Vanderburg, 9459 Taylorsville Rd, Dayton OH 45424. A membership fee of $6 includes a 6 issue subscription. Back issues are available for $1 apiece.

Permian Basin Computer Group — Midland and Odessa Colleges TX

The Permian Basin Computer Group has factions on both the Midland and Odessa College campuses. The Midland group meets on the second Monday of the month at 7:30 in the Student Union Building. The Odessa group meets on the second Saturday at 1:30 in the Electronic Technology Building, Room 203. For additional information write John Raenaldt, POB 3912, Odessa TX or phone (915) 332-9151 Mondays through Fridays between 9:00 and 5:00 or (915) 697-4607 after 6 PM.

Goodyear Computer Club

The Goodyear Computer Club in Akron OH is off to an auspicious start: just a few months old and able to boast a membership of more than 150. Generally the members are associated with the Goodyear Aerospace Corporation or the Goodyear Tire and Rubber Company. Contact the Goodyear Computer Club c/o J F Derry, D-109 Plt 1, The Goodyear Tire and Rubber Co, Akron OH 44316.

Aloha Computer Club

Don Henson, president of the Aloha Computer Club of Millani Town HI, has made an interesting offer. He's willing to give monthly seminars limited to six beginners. Sounds like the best possible way to share computer information and get more of

SERIAL SYSTEMS MODEL 301

(The Black Box)

Max performance at Minimum price—Serial data to FSK of FSK to Serial Communicate between uProcessor and cassette tape, or radio transceiver and RTTY, also easily adapted to phone—50 to 600 BAUD—Crystal controlled—Programmable for all standard freq's (Kansas City, Altair, Amature RTTY)

Assy and Tested $49.95

New 90 Key keyboard Encoder Chip Hook Row and Column to your keyboard and select ASCII, BAUDOT, ANSI TTY, or ASR-33 outputs.
$24.50

TMS9900 16 bit uP Chip $99.50

Serial Systems (Div. of ASC)
P.O. Box 20330
Minneapolis, MN 55420

Circle 69 on inquiry card.

NOW HEAR THIS!!

ALPHA DIGITAL SYSTEMS

SELLS IMSAI KITS AND TOTAL INTEGRATED SYSTEMS ASSEMBLED, CHECKED-OUT, AND WARRANTED AT KIT PRICES.

ALPHA I

SPECIAL DOS SYSTEM DEAL (includes)
* IMSAI 8080 Computer
* 22 Slot Mother Bd. with Conn.
* 16 K RAM
* 90 K Disk (with Controller)
* Selectric typewriter (with Controller)
* DOS-BASIC Software

Assembled System Total $3995.
(Less Selectric) $2495

Select any kit or system of kits from the IMSAI price list, order from us and receive the assembled unit for the same price. If you don't have a price list, drop us a line and we will send you one. Terms: Cash with Order - Prices include freight. (N.C. Residents Add 4% Sales Tax)

How can ALPHA DIGITAL SYSTEMS do all this? Its simple. ALPHA DIGITAL WANTS TO BE YOUR COMPUTER COMPANY.

ADS ALPHA DIGITAL SYSTEMS
RT. 4 BOX 171A
BOONE, N.C. 28607
(704) 264-7946

Circle 70 on inquiry card.

PHOENIX
Bits & Bytes
COMPUTER SHOP

- Lear Siegler ADM-3 Kit $849.
- Vector-1 8080 16 K $968.
- 9" Sanyo Monitor $159.

North Star - EPA - Tarbell
Act 1 Terminals - Cybercom
IASIS Computer-in-a-book
Vector Graphics - Per Com

6819-C N. 21st Ave.
Phoenix, Az. 85015
(602) 242-2507

Circle 71 on inquiry card.

SAVE Money, & Time. Send $5.82 for 12 issues of an INDEPENDENT newsletter on how HEATHKITs™ can help you use computers.
Name:
Address:
charles Floto
267 Willow St. 'B'
NewHaven CT 06511

Heathkit is a registered trademark of Heath Company.

Circle 72 on inquiry card.

CHEAP, INC.

(301) 779-7998
7338 Baltimore Ave., Suite 200 College Park, Maryland 20740

IMSAI made kits & assembled units	17% OFF LIST
TEC-9900-SS-UF, 16 bit TI9900 microP, 32 bit I/O hardware mult & div, buffered bus, 20 ma or RS232, 8 interrupts & sockets	$279.00
TEC-9900-32KB 32K bytes memory	$649.00
TEC-9900-PP power supply	$129.00
CROMEMCO kits & Z-1,Z-2	10% OFF LIST
SANYO TV monitors	10% OFF LIST
TDL ZPU & Z16K	12% OFF LIST
SEALS 8K bat. backup Mem.	10% OFF LIST
PERSCI Dual floppy & intelligent controller (#270 & 1070)	5% OFF LIST
SHUGART MINI-FLOPPY	$375.00

21L02 500 ns	$1.90	2102 500 ns	$1.85
21L02 450 ns	$1.50	2102 400 ns	$1.95
21L02 400 ns	$1.99	Z-80 PIO	$9.50
Z-80 microP	$45.00	1702A EPROM	$8.00
Z-80 CTC	$9.50	8080A 30 MH	$20.00
8080A 2.5 MH	$17.00	8080 SCI	$10.00
8080 PPI	$10.00	8224 Clock Gen.	$4.75
8212 I/O port	$2.50	8216/26 Driver	$2.50
8228 Clock Driver	$5.50	8251 PCI	$9.50
8255 PPI	$9.50	2102 650 ns	$1.75
2708 EPROM	$29.00		

ADM-3A +CRT & addressable cursor $925.00 list	$799.00
Nat. Multiplex Digital Tape recorders & kits	10% OFF LIST
MSI floppy & kits	10% OFF LIST
Pennywhistle modem	5% OFF LIST
TI "silent 700" model 743, printer $1395 list	5% OFF LIST

Send $2.00 for newsletter and list. 4%, with minimum $4.00 for shipping and handling. See flyer for policies and delivery times on group OEM buys and other products.

Circle 73 on inquiry card.

IT'S A GREAT BIG COMPUTER WORLD
But You Only Need
THE COMPUTER CORNER

- SOL – A New Dawn Is Here!
- IMSAI 8080
- POLY – 88
- TDL Z-80
- Memories & I/O Boards
- Computer Book Service
- Magnetic Tapes & Disks
- Full Line of Magazines
- Brain Games & Puzzles
- Workshops & Club Information

Visit THE COMPUTER CORNER for all your computer needs. Stop in and browse – you'll like our personal service.

THE COMPUTER CORNER
White Plains Mall – Upper Level
200 Hamilton Avenue
White Plains, New York 10601
Tel: (914) WHY - DATA

Ample Parking
10-6 Daily & Saturday
10-9 Thursday

Circle 74 on inquiry card.

computer depot inc.™

Parts galore in the Midwest for

PROCESSOR TECHNOLOGY, IMSAI, POLYMORPHIC SYSTEMS, DIGITAL GROUP, WAVE MATE, CROMEMCO, VECTOR GRAPHICS, TDL, SEALS, TARBELL, MICRO DESIGNS, NORTH STAR, PERIPHERAL VISION; INTEL, NATIONAL SEMICONDUCTOR, FAIRCHILD SEMICONDUCTOR, MOSTEK, E & L INSTRUMENTS, SAMS, TAB, HAYDEN, WILEY, MC-GRAW HILL

Catalog Free

Computer Depot Inc.
3515 W. 70th Street
Minneapolis MN 55435 C40

Circle 75 on inquiry card.

IN
SOUTH FLORIDA
IT'S

COMPUTER HUT

FEATURING POLY 88

5905 NW 151 St.
Miami Lakes, FL 33017
(305) 558-8080

Circle 76 on inquiry card.

COMPUTER MART OF

DEALERS FOR
TDL ICOM IMSAI
DIGITAL GROUP

SOFTWARE INCLUDES:

8K BASIC EXT. BASIC
TEXT EDITOR DISK BASIC
WORD PROCESSOR
MACRO-ASSEMBLER

170 MAIN STREET NASHUA
(603) 883-2386

Circle 77 on inquiry card.

the many, many interested people involved in personal computing. Aloha meets on the first Wednesday of the month at 7:30 at the Waikiki-Kapahulu Library, 400 Kapahulu Av. Seminars are held at various times throughout the month. New members are more than welcome. To learn more write 94-360 Hokuala St, Apt 187, Millani Town HI 96789 or call (808) 623-1781.

Central Florida Computer Club

A new hobbyist group has been formed in Orlando. Jim Walton is the contact person for the Central Florida Computer Club. He can be reached c/o Data Entry Engineering, 1810 N Orange Av, Orlando FL 32804, (305) 896-4322.

Electronotes — Newsletter of the Musical Engineering Group

Those people interested in both electronic music and instrument construction will find *Electronotes* most valuable. An in-depth presentation focusing on the ENS-76 is planned for the next several issues. *Electronotes* is published by B A Hutchins, 203 Snyder Hill Rd, Ithaca NY 14850. Routine orders should be sent c/o *Electronotes*, 213 Dryden Rd, Ithaca NY 14850.

Pittsburgh Area Computer Club

The place to share computer lore in Western Pennsylvania is the Pittsburgh Area Computer Club. Contact PACC at 400 Smithfield St, Pittsburgh PA 15222.

A New Orleans Computerfest

The Jefferson Amateur Radio Club and the Crescent City Computer Club would like to announce the New Orleans Hamfest/Computerfest which will be held at the Hilton Inn in Kenner LA (directly across from the New Orleans International Airport) September 24 and 25. This is the ARRL Delta Division Convention for 1977 and is the largest "ham" outing in the deep south.

This year's event will feature a banquet Saturday night with entertainment, two days of commercial exhibits, flea markets and forums. There will also be a hospitality room, ladies' events, FCC examinations and more.

This year's grand prize is a complete Drake "C-Line" ham station, and many door prizes will be awarded each day.

Information on tickets, room reservations, etc, will be furnished upon request by contacting the New Orleans Hamfest/Computerfest, POB 10111, Jefferson LA 70181.

Important Notice to Clubs

Hazeltine Corporation, Greenlawn NY 11740, has recently circulated a letter from Frank J Cirillo of their Industrial Products Division, announcing a special package price of $1400 for refurbished H-2000 model terminals with Hazeltine tape cassette drive. The letter is directed to computer clubs, so if your local club did not receive a copy, write or call Mr Cirillo at (516) 261-7000 and inquire about this excellent deal. This terminal is one of the world's largest selling video display terminals and has had application in universities, hopitals, business, finance and government. At a $1400 price you get a used commercial grade piece of equipment which will prove quite applicable to many home computer situations.■

Would your club benefit from a write-up in BYTE's Clubs and Newsletters section? It's easy enough to arrange. Just drop a line to BYTE, 70 Main St, Peterborough NH 03458, c/o Clubs and Newsletters, with the information you'd like published. Naturally we can't cover every club every month but we do make a special effort to mention new clubs and those that correspond regularly.

Those people looking for a more complete listing of computer clubs should refer to January 1977 BYTE.■

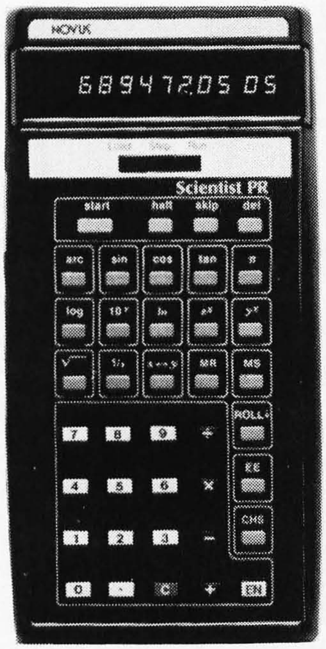
Programming Quickies

BASIC Timing Delay

Gregory A Worth
115 Campbell, Apt 6
Rochester MI 48063

When writing game programs it is often convenient to have a time delay to add some exciting real life feeling to the game. This subroutine, submitted by Gregory A Worth and written for a Motorola 6800, will cause delays between 1 and 255 seconds. Register B is loaded with the number of seconds you wish to delay before jumping to the subroutine. LOOP1 is the basic time delay loop. This loop takes 20 clock cycles and is executed 50000 times giving a basic timing unit of 1 second, assuming a 1 MHz processor clock is used. If your system is using a different clock speed, appropriate adjustment of either the number of repetitions of the loop or the amount of time consumed in one pass through the loop will have to be made. The outer loop, LOOP2, allows the varying time delays by repeating the 1 second loop the number of times specified in register B. All of the numbers in this listing are in hexadecimal unless otherwise specified.

Address	Hex	Code	Labels	Op	Operand	Commentary
0000	FF	00 16	DELAY	STX	SAVE	save the number of seconds to be delayed;
0003	CE	C3 50	LOOP2	LDX	C350	load X register with decimal 50000;
0006	76	00 03	LOOP1	ROR	LOOP2	} basic 1 second timing loop considering 1 MHz clock;
0009	79	00 03		ROL	LOOP2	
000C	09			DEX		
000D	26	F7		BNE	LOOP1	
000F	5A			DECB		decrement number of seconds delayed;
0010	26	F1		BNE	LOOP2	if delay not over continue LOOP2;
0012	FE	00 16		LDX	SAVE	restore value of delay time;
0015	39			RTS		return to calling program;
0016			SAVE	RMB	2	storage for number of seconds delayed;
0018				END■		



SPECIAL OFFER 100 STEPS PROGRAMMABLE SCIENTIST

#4525

REG. $89.95

$29.99

· RPN logic with "built-in" hierarchy for increased accuracy and speed in calculating sequences involving arithmetic, trigonometric, logarithmic, power or exponential functions. · A rollable 4-level stack lets you review or use intermediate solutions. · Eight-digit plus 2-digit exponent LED display with full-floating decimal system. · Scientific notation for increased mathematical capacity. · Sine, cosine, tangent and inverse trigonometric functions. · Common and natural logarithms and antilogarithms. · Instant automatic calculation of powers and roots. · Single-key square root calculations. · Single-key Pi entry. · Separate storage memory. Square, square root and reciprocal calculations. Change sign and register exchange keys. · MOS/LSI solid-state circuitry. · Includes 3 AA rechargable NiCad batteries.

**MANUFACTURED IN USA BY
BY NATIONAL SEMICONDUCTOR
ONE YEAR UNLIMITED WARRANTY
10 DAYS MONEY BACK GUARANTEE**

____Model #4525@38.95 ____
____Model #4520@28.88 ____
____AC Charger@4.95 ____
____Case(s)@2.95 ____
____Calc. Stand@2.50 ____
Cal. Res. add 6% tax ____

Total ____

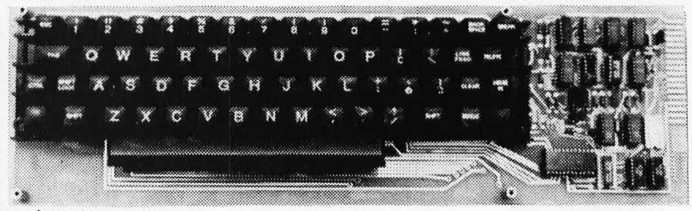
ALSO AVAILABLE

MODEL #4520 With the same features as above but without programming capability @ $28.88. Accessories same price as above.

Circle 157 on inquiry card.

SAME DAY SERVICE IF PAID BY CASHIER'S CHECK, MONEY ORDER OR CHARGE CARD.

ASC II KEYBOARD
NEW LOOK IMPROVED DESIGN



$58.00

This 63 key **ASC II Encoded Keyboard** kit was designed and manufactured by **Electronics Warehouse Inc.** Features: Single 5 volt D.C. supply, utilizing only TTL Logic elements (no MOS devices to blow). TTL drive capability (each of the eight bits of ASC II output will drive the equivalent of ten standard TTL inputs without external buffer drivers), de-bouncing, upper and lower case fully ASC II, 8 bit parallel. In addition to the alpha-numeric and symbol keys available on a regular keyboard, the following keys are utilized: Escape, back-space, tab, line-feed, delete, control, shift-lock, shift (2 keys), return. All 128 ASC II characters are generated.

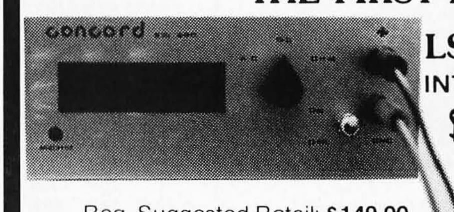
Kit includes: 63 key keyboard, P.C. board, all required components and assembly manual with ASC II code list.

Optional: Parity bit - add 50¢ • Enclosure - $25.00 • Serial output - add $2.00 • 18 Pin edge connector - $2.00 • Sockets - $4.00

Note: If you already have this teletype keyboard you can have the kit without it for **$39.00.** Dealer inquiries invited.

FROM CONCORD

THE FIRST FULL *FEATURE* LSI DMM KIT



INTRODUCTORY PRICE:

$77.77

Reg. Suggested Retail: $149.00

- AUTO RANGING
- AUTO POLARITY
- AUTO ZERO
- 3 Large Digits (1/2")
- Rechargable

MEASUREMENT RANGES:

Voltage: (AC & DC) 1 MV - 1000V
Current: (AC & DC) 10 μA - 1A
RESISTANCE: 1 Ω - 10 M Ω
Basic D.C. Accuracy, better than 0.1% ± 1 Digit
Power: 4 AA batteries (Rechargable batteries optional)

NI-CAD BATTERIES: $6.00 • AC CHARGER: $4.95 • ENCLOSURE: $12.95 • TEST LEADS: $1.95 • SHUNT KIT FOR 3 CURRENT RANGES: $4.75 • SOCKETS $2.50

ORDERING INFORMATION

SHIPPING AND HANDLING - $3.00 + 50¢ Insurance
California residents add 6% sales tax

ELECTRONICS WAREHOUSE Inc.

**1603 AVIATION BLVD. Dept. B
REDONDO BEACH, CA. 90278
TEL. (213) 376-8005**

WRITE FOR FREE CATALOG

You are invited to visit our store at the above address

Circle 85 on inquiry card.

COMPUTERS: circle desired kits, then show this ad to your owner.

1 Active Terminator Board $25

- Plugs into your S-100 bus to actively terminate lines
- Minimize the crosstalk, noise, overshoot, and ringing you can find on improperly terminated busses

2 MOTHERBOARD(S) 10 Slots: $85 18 Slots: $118

- Prices include edge connectors
- Includes active terminations for unambiguous data transfer
- Uses glass epoxy, plate-through, double sided board

3 8K EconoRAM II T.M. $163.84

- Low current drain, zero wait states (1 wait with Z-80)
- Tri-state outputs mate with any type of bus
- Configured as two separate 4K blocks (with fully independent protect and address decoding to increase flexibility)
- Includes low power Schottky support ICs and full buffering

4 4K EconoROM T.M. $265

- NEW AND IMPROVED---better listings; SP and PSW now implemented for more flexibility. Find out why Jay Bell, editor of Print-Out, said:

  "Without a doubt, this board is the best buy I have made for my Altair 8800 out of the $5000+ I've spent." (Print-Out, September 1976 issue)

  Now you can have editor, assembler, and monitor routines that make the 8080 a much nicer chip to work with.

5 CPU Power Supply $45

- 5 Volts @ 4 Amps with crowbar overvoltage protection
- +12 Volts @ ½ Amp and -12 Volts @ ½ Amp
- Adjustable bias supply, 5 to 10 Volts at 10 mA
- Intended for use with small computer systems or as digital bench supply

CAVE graphix

GODBOUT

BILL GODBOUT ELECTRONICS
BOX 2355, OAKLAND AIRPORT, CA 94614

TERMS: Please allow up to 5% for shipping; excess refunded. Californians add sales tax. Street address must be included for COD orders. Place BankAmericard® and Mastercharge® orders ($15 minimum) by calling our 24 hour order desk at (415) 562-0636.

FREE FLYER: Just give us the word and we'll send out a copy of our flyer, which describes our complete line of products in greater detail.

Circle 86 on inquiry card.

7400N TTL

SN7400N	.16	SN7459A	.25		
SN7401N	.16	SN7460N	.22	SN74154N	1.00
SN7402N	.21	SN7470N	.45	SN74155N	.99
SN7403N	.16	SN7472N	.39	SN74156N	.99
SN7404N	.18	SN7473N	.37	SN74157N	.99
SN7405N	.24	SN7474N	.32	SN74160N	1.25
SN7406N	.20	SN7475N	.50	SN74161N	.99
SN7407N	.29	SN7476N	.32	SN74163N	.99
SN7408N	.25	SN7479N	5.00	SN74164N	1.10
SN7409N	.25	SN7480N	.50	SN74165N	1.10
SN7410N	.18	SN7482N	.98	SN74166N	1.25
SN7411N	.30	SN7483N	.70	SN74167N	5.50
SN7412N	.33	SN7485N	.89	SN74170N	2.10
SN7413N	.45	SN7486N	.39	SN74172N	8.95
SN7414N	.70	SN7488N	3.50	SN74173N	1.50
SN7416N	.35	SN7489N	2.49	SN74174N	1.25
SN7417N	.35	SN7490N	.45	SN74175N	.99
SN7420N	.21	SN7491N	.75	SN74176N	.90
SN7421N	.33	SN7492N	.49	SN74177N	.90
SN7422N	.49	SN7493N	.49	SN74180N	.99
SN7423N	.37	SN7494N	.79	SN74181N	2.49
SN7425N	.29	SN7495N	.79	SN74182N	.95
SN7426N	.29	SN7496N	.89	SN74184N	1.95
SN7427N	.37	SN7497N	4.00	SN74185N	2.20
SN7429N	.42	SN74100N	1.00	SN74186N	15.00
SN7430N	.26	SN74107N	.39	SN74187N	6.00
SN7432N	.31	SN74121N	.39	SN74188N	3.95
SN7437N	.27	SN74122N	.39	SN74190N	1.19
SN7438N	.27	SN74123N	.50	SN74191N	1.25
SN7439N	.25	SN74125N	.60	SN74192N	.89
SN7440N	.15	SN74126N	.60	SN74193N	.89
SN7441N	.89	SN74132N	1.09	SN74194N	1.25
SN7442N	.59	SN74136N	.95	SN74195N	.75
SN7443N	.75	SN74141N	1.15	SN74196N	1.25
SN7444N	.75	SN74142N	4.00	SN74197N	.75
SN7445N	.75	SN74143N	4.50	SN74198N	1.75
SN7446N	.81	SN74144N	4.50	SN74199N	1.75
SN7447N	.69	SN74145N	1.15	SN74200N	5.59
SN7448N	.79	SN74147N	2.35	SN74279N	.90
SN7450N	.26	SN74148N	2.00	SN74251N	1.79
SN7451N	.27	SN74150N	1.00	SN74284N	6.00
SN7453N	.27	SN74151N	.79	SN74285N	6.00
SN7454N	.20	SN74153N	.89	SN74367N	.75

MANY OTHERS AVAILABLE ON REQUEST
20% Discount for 100 Combined 7400's

CMOS

CD4000	.25			74C04N	.75
CD4001	.25			74C10N	.65
CD4002	.25	CD4035	1.85	74C20N	.65
CD4006	2.50	CD4040	2.45	74C30N	.65
CD4007	.25	CD4042	1.90	74C42N	2.15
CD4009	.59	CD4044	1.50	74C73N	1.50
CD4010	.59	CD4046	2.51	74C74	1.15
CD4011	.25	CD4047	2.75	74C90N	3.00
CD4012	.25	CD4049	.79	74C95N	2.00
CD4013	.47	CD4050	.79	74C107N	1.25
CD4016	.56	CD4051	2.95	74C151	2.90
CD4017	1.35	CD4053	2.95	74C154	4.00
CD4019	.55	CD4060	3.25	74C157	2.15
CD4020	1.49	CD4066	1.75	74C160	3.25
CD4022	1.25	CD4069	.45	74C161	3.25
CD4023	.25	CD4071	.45	74C163	3.00
CD4024	1.50	CD4081	.45	74C164	3.25
CD4025	.25	CD4511	2.50	74C173	2.60
CD4027	.69	CD4518	2.50	74C193	2.75
CD4028	1.65	MC14566	3.00	74C195	2.75
CD4029	2.90	74C00N	.39	MC4044	4.50
CD4030	.65	74C02N	.55	MC14016	.56

LINEAR

LM300H	.80			LM1351N	1.65
LM301H	.35			LM1414N	1.75
LM301CN	.35	78MG	1.75	LM1458C	.65
LM302H	.75	LM370N	1.1.5	LM1496N	.95
LM304H	1.00	LM373N	3.25	LM1556V	1.85
LM305H	.95	LM377N	4.00	LM2111N	1.95
LM307CN	.35	LM380N	1.39	LM2901N	2.95
LM308H	1.00	LM380CN	1.05	LM3065N	.69
LM308CN	1.00	LM381N	1.79	LM3900N	.55
LM309H	1.10	LM382N	1.79	LM3905N	.60
LM309K	.99	NE501K	8.00	LM3909	1.25
LM310CN	1.15	NE510A	6.00	LM5556N	1.85
LM311H	.90	NE531H	3.00	MC5558V	1.00
LM311N	.90	NE536T	6.00	LM7525N	.90
LM318CN	1.50	NE540L	6.00	LM7535N	1.25
LM319N	1.30	NE550N	.79	8038B	4.95
LM320K-5	1.35	NE555V	.39	LM75450	.49
LM320K-5.2	1.35	NE560B	5.00	75451CN	.39
LM320K-12	1.35	NE561B	5.00	75452CN	.39
LM320K-15	1.35	NE562B	5.00	75453CN	.39
LM320T-5	1.75	NE565H	1.25	75454CN	.39
LM320T-5.2	1.75	NE565N	1.75	75491CN	.79
LM320T-8	1.75	NE566CN	1.25	75492CN	.89
LM320T-12	1.75	NE567H	1.95	75494CN	.89
LM320T-15	1.75	NE567V	1.50	**RCA LINEAR**	
LM320T-18	1.75	LM703CN	.45	CA3013	2.15
LM320T-24	1.75	LM709H	.29	CA3023	2.56
LM323K-5	9.95	LM709N	.29	CA3035	2.48
LM324N	1.80	LM710N	.79	CA3039	1.35
LM339N	1.70	LM711N	.39	CA3046	1.30
LM340K-5	1.95	LM723H	.55	CA3059	3.25
LM340K-6	1.95	LM723H	.55	CA3060	3.25
LM340K-8	1.95	LM733N	1.00	CA3080	.85
LM340K-12	1.95	LM739N	1.00	CA3081	2.00
LM340K-15	1.95	LM741CH	.35	CA3082	2.00
LM340K-18	1.95	LM741CN	.35	CA3083	1.60
LM340K-24	1.95	LM741-14N	.39	CA3086	.85
LM340T-5	1.75	LM747H	.79	CA3089	3.75
LM340T-6	1.75	LM747N	.79	CA3091	10.20
LM340T-8	1.75	LM748H	.39	CA3102	2.95
LM340T-12	1.75	LM748N	.39	CA3123	2.15
LM340T-15	1.75	LM1303N	.90	CA3130	1.39
LM340T-18	1.75	LM1304N	1.19	CA3140	1.25
LM340T-24	1.75	LM1305N	1.40	CA3600	1.75
LM350N	1.00	LM1307N	.85	RC4194	5.95
LM351CN	.65	LM1310N	2.95	RC4195	3.25

74LS00 TTL

74LS00	29			74LS139	1.95
74LS02	29			74LS151	1.55
74LS03	29	74LS74	49	74LS153	1.89
74LS04	35	74LS75	69	74LS157	1.55
74LS05	35	74LS76	49	74LS162	2.25
74LS08	29	74LS83	1.75	74LS163	2.25
74LS10	29	74LS85	2.49	74LS164	1.95
74LS13	69	74LS86	49	74LS175	1.95
74LS14	1.75	74LS90	1.25	74LS181	3.69
74LS20	29	74LS92	1.25	74LS190	2.85
74LS26	39	74LS93	1.25	74LS191	2.85
74LS27	39	74LS95	1.95	74LS192	2.85
74LS28	39	74LS96	1.89	74LS193	2.85
74LS30	29	74LS107	59	74LS194	1.89
74LS32	39	74LS109	59	74LS195	1.89
74LS40	39	74LS112	59	74LS257	1.75
74LS51	29	74LS132	1.25	74LS260	55
74LS55	29	74LS136	59	74LS279	79
74LS73	49	74LS138	1.89	74LS670	3.95

CLOCK CHIPS

MM5309	6 Digit, BCD Outputs, Reset PIN.	$9.95
MM5311	6 Digit, BCD Outputs, 12 or 24 Hour	4.95
MM5312	4 Digit, BCD Outputs, 1 PPS Output	4.95
MM5314	6 Digit, 12 or 24 Hour, 50 or 60 Hz	4.95
MM5316	4 Digit, Alarm, 1 PPS Output	6.95
MM5318	Video Clock Chip, For Use With (MM5841 - $9.95)	9.95
CT7001	6 Digit, Calendar, Alarm, 12 or 24 Hour	5.95

DATA HANDBOOKS

7400	Pin-out & Description of 5400/7400 ICS	$2.95
CMOS	Pin-out & Description of 4000 Series ICS	$2.95
Linear	Pin-out & Functional Description	$2.95

ALL THREE HANDBOOKS $6.95

Timeband™ by FAIRCHILD — Watches —



T201 Black Bracelet LED $19.95



T237 White w/bracelet $29.95
T236 Yellow w/bracelet $34.95
LED

Men's & Ladies

- Solid State
- Displays hour, minute, second, month & day
- Snap-out battery replacement
- Free set of replacement batteries
- Choose LED or LCD styles
- One year factory warranty



TC441 White w/strap $29.95
TC440 Yellow w/strap $34.95
LCD

LADIES LED



T311 White w/strap $34.95
T310 Yellow w/strap $39.95

CHANNEL F™ FAIRCHILD VIDEO ENTERTAINMENT SYSTEM



$159.95

- Freeze Action • Speed Option
- Automatic time and scorekeeping
- Battery-free AC operation
- Dual controls with 8-way action
- Built-in Pro Hockey and Tennis games
- Easy hook-up on any B/W or Color TV
- Factory warranty

Channel F — additional cartridges — $19.95 ea.

#8112 - Desert Fox/Shooting Gallery
#8113 - Blackjack (1 or 2 players)
#8114 - Spitfire (1 or 2 players)
#8115 - Space War
#8111 - Tic-Tac-Toe/Shooting Gallery/Doodle/Quadra-Doodle

DISCRETE LEDS

125" dia.			.190" dia.		
XC209	Red	10/$1	XC111	Red	10/$1
XC209	Green	4/$1	XC111	Green	4/$1
XC209	Orange	4/$1	XC111	Yellow	4/$1
XC209	Yellow	4/$1	XC111	Orange	4/$1

.200" dia			.185" dia.			.200" dia.		
XC22	Red	10/$1	XC526	Red	5/$1	XC556	Red	10/$1
XC22	Green	4/$1	XC526	Green	4/$1	XC556	Green	4/$1
XC22	Yellow	4/$1	XC526	Yellow	4/$1	XC556	Yellow	4/$1
XC22	Orange	4/$1	XC526	Orange	4/$1	XC556	Orange	4/$1
SSL-22	RT	4/$1	XC526	Clear	4/$1	XC556	Clear	7/$1

.085" dia. MV50 - Red - 6/$1

INFRA-RED LED ¼" x ¼" x 1/16" Flat **5/$1.00**

SPECIAL * — XC556 Red 100/$8.00 1000/$60.00 — SPECIAL *

DISPLAY LEDS

DL707 DL33B

TYPE	POLARITY	HT		TYPE	POLARITY	HT	
MAN 1	Common Anode	270	2.95	MAN 3640	Common Cathode-orange	300	1.75
MAN 2	5 x 7 Dot Matrix	300	4.95	MAN 4710	Common Anode-Red	.400	1.95
MAN 3	Common Cathode	125	3/1.00	DL701	Common Anode-red ±	300	.99
MAN 4	Common Cathode	187	1.95	DL704	Common Cathode	.300	.99
MAN 7	Common Anode	300	1.25	DL707	Common Anode	.300	.99
MAN 7G	Common Anode-green	300	1.95	MAN 4740	Common Anode-Red	.400	.99
MAN 7Y	Common Anode-yellow	300	1.95	DL741	Common Anode	.600	1.50
MAN 52	Common Anode-green	300	.99	DL 747	Common Anode	600	2.25
MAN 64	Common Anode-red	.400	.99	DL 750	Common Cathode	600	2.49
MAN 74	Common Cathode	300	1.50	DL 338	Common Cathode	110	.50
MAN 82	Common Anode-yellow	300	.99	FND70	Common Cathode	250	.75
MAN 84	Common Cathode-yellow	300	.99	FND503	Common Cathode	500	1.00
MAN 3620	Common Anode -orange	300	1.75	FND507	Common Anode	.500	1.00

FACTORY REJECTS ATARI GAME BOARDS

BOARD A — 8½" x 16" — **$6.95 ea.** ONLY 500 EA AVAILABLE
Over 60 each reusable IC's
Misc. Transistors, Resistors, Diodes, Caps, Crystals, Switch, etc.

DELUXE BOARD B — 11½" x 18" — **$9.95 ea.** ONLY 500 EA. AVAILABLE
Over 100 each reusable IC's
Misc. Tranistors, Resistors, Diodes, Caps, Crystals, Switches, LEDS, etc.

HP 5082-7300 Multi-Digit Series

- ⅛" Ht. • Common Cathode • Dip Package
- 3 to 5 volts @ 5 mils per segment
- 7 segment Monolithic • Red Display



2 Digit	$ .79
3 Digit	.89
4 Digit	.99
5 Digit	1.19

IC SOLDERTAIL — LOW PROFILE (TIN) SOCKETS

	1-24	25-49	50-100		1-24	25-49	50-100
8 pin	$.17	16	15	24 pin	$.38	.37	.36
14 pin	.20	19	.18	28 pin	.45	.44	43
16 pin	.22	21	.20	36 pin	60	59	.58
18 pin	.29	28	.27	40 pin	.63	.62	61
22 pin	.37	.36	.35				

SOLDERTAIL STANDARD (TIN)

14 pin	$.27	.25	.24	28 pin	$ .99	.90	.81
16 pin	.30	.27	.25	36 pin	1.39	1.26	1.15
18 pin	.35	.32	30	40 pin	1.59	1.45	1.30
24 pin	.49	.45	42				

SOLDERTAIL STANDARD (GOLD)

8 pin	$.30	27	24	24 pin	$. 70	.63	.57
14 pin	.35	32	.29	28 pin	1.10	1.00	90
16 pin	.38	35	32	36 pin	1.75	1.40	1.26
18 pin	52	47	43	40 pin	1.75	1.59	1.45

WIRE WRAP SOCKETS (GOLD) LEVEL #3

10 pin	$.45	.41	.37	24 pin	$1.05	.95	.85
14 pin	.39	.38	37	28 pin	1.40	1.25	1.10
16 pin	.43	.42	41	36 pin	1.59	1.45	1.30
18 pin	.75	68	62	40 pin	1.75	1.55	1.40

50 PCS. RESISTOR ASSORTMENTS $1.25 PER ASST.

SPECIAL

ASST. 1	5 ea.	10 OHM 27 OHM	12 OHM 33 OHM	15 OHM 39 OHM	18 OHM 47 OHM	22 OHM 56 OHM	1/4 WATT 5% = 50 PCS.
ASST. 2	5 ea.	68 OHM 180 OHM	82 OHM 220 OHM	100 OHM 270 OHM	120 OHM 330 OHM	150 OHM 390 OHM	1/4 WATT 5% = 50 PCS.
ASST. 3	5 ea.	470 OHM 1.2K	560 OHM 1.5K	680 OHM 1.8K	820 OHM 2.2K	1K 2.7K	1/4 WATT 5% = 50 PCS.
ASST. 4	5 ea.	3.3K 8.2K	3.9K 10K	4.7K 12K	5.6K 15K	6.8K 18K	1/4 WATT 5% = 50 PCS.
ASST. 5	5 ea.	22K 56K	27K 68K	33K 82K	39K 100K	47K 120K	1/4 WATT 5% = 50 PCS.
ASST. 6	5 ea.	150K 390K	180K 470K	220K 560K	270K 680K	330K 820K	1/4 WATT 5% = 50 PCS.
ASST. 7	5 ea.	1M 2.7M	1.2M 3.3M	1.5M 3.9M	1.8M 4.7M	2.2M 5.6M	1/4 WATT 5% = 50PCS.

ASST. 8R Includes Resistor Assortments 1-7 (350 PCS.) **$7.49 ea.**

SPECIAL

$5.00 Minimum Order — U.S. Funds Only
California Residents — Add 6% Sales Tax
Spec Sheets - 25¢ — Send 35¢ Stamp for 1977A Catalog
Dealer Discount Available — Request Pricing

NOW OPEN SATURDAYS

James ELECTRONICS

1021-A HOWARD AVE., SAN CARLOS, CA. 94070
PHONE ORDERS WELCOME — (415) 592-8097
All Advertised Prices Good Thru July

WIRE WRAP CENTER

HOBBY-WRAP TOOL-BW-630



- Battery Operated (Size C)
- Weighs ONLY 11 Ounces
- Wraps 30 AWG Wire onto Standard DIP Sockets (.025 inch)
- Complete with built-in bit and sleeve

$34.95
(batteries not included)

WIRE-WRAP KIT — WK-2-W

WRAP • STRIP • UNWRAP



- Tool for 30 AWG Wire
- Roll of 50 Ft. White or Blue 30 AWG Wire
- 50 pcs. each 1", 2", 3" & 4" lengths — pre-stripped wire.

$11.95

WIRE WRAP TOOL WSU-30



WRAP • STRIP • UNWRAP - $5.95

WIRE WRAP WIRE — 30 AWG

25 ft. min. $1.25 50 ft. $1.95 100 ft. $2.95 1000 ft. $15.00
SPECIFY COLOR — White - Yellow - Red - Green - Blue - Black

Plastic Push Button Switch



- 18 AWG Solid Wire - 5" Long
- .50 (wide) X .60 (high) ⅜-27 Thread
- 8 AMP @ 14 Volt - 1 AMP @ .10 Volt

		1-9	10-Up
J-188-1	Push On-Push Off	.59	.49
J-188-2	Normally Open	.59	.49
J-188-3	Normally Closed	.59	.49

DIP SWITCHES

SPST Slide Action



#206-4	(8 pin dip)	4 switch unit	**$1.75 ea.**
#206-7	(14 pin dip)	7 switch unit	**$1.95 ea.**
#206-8	(16 pin dip)	8 switch unit	**$2.25 ea.**

TV GAME CHIP SET — $18.50

Includes AY-3-8500-1 chip and 2.010 mhz crystal — if purchased separately would cost $21.90.

ZENERS — DIODES — RECTIFIERS

TYPE	VOLTS	W	PRICE	TYPE	VOLTS	W	PRICE
1N746	3.3	400mm	4/1.00	1N4005	600 PIV	1 AMP	10/1.00
1N751A	5.1	400m	4/1.00	1N4006	800 PIV	1 AMP	10/1.00
1N752	5.6	400m	4/1.00	1N4007	1000 PIV	1 AMP	10/1.00
1N753	6.2	400m	4/1.00	1N3600	50	200m	6/1.00
1N754	6.8	400m	4/1.00	1N4148	75	10m	15/1.00
1N959	8.2	400m	8/1.00	1N4154	35	10m	12/1.00
1N965B	15	400m	4/1.00	1N4305	75	25m	20/1.00
1N5232	5.6	500m	28	1N4734	5.6	1w	28
1N5234	6.2	500m	28	1N4735	6.2	1w	28
1N5235	6.8	500m	28	1N4736	6.8	1w	28
1N5236	7.5	500m	28	1N4738	8.2	1w	28
1N456	25	40m	6/1.00	1N4742	12	1w	28
1N458	150	7m	6/1.00	1N4744	15	1w	28
1N485A	180	10m	6/1.00	1N1183	50 PIV	35 AMP	1.60
1N4001	50 PIV	1 AMP	12/1.00	1N1184	100 PIV	35 AMP	1.70
1N4002	100 PIV	1 AMP	12/1.00	1N1185	150 PIV	35 AMP	1.50
1N4003	200 PIV	1 AMP	12/1.00	1N1186	200 PIV	35 AMP	1.80
1N4004	400 PIV	1 AMP	12/1.00	1N1188	400 PIV	35 AMP	3.00

SCR AND FW BRIDGE RECTIFIERS

C36D	15A @ 400V	SCR	$1.95
C38M	35A @ 200V	SCR	1.95
2N2328	1.6A @ 200V	SCR	.50
MDA 980-1	12A @ 50V	FW BRIDGE REC.	1.95
MDA 980-3	12A @ 200V	FW BRIDGE REC.	1.95

TRANSISTORS

MPS A05	5/$1.00			PN4249	4/$1.00
MPS A06	5/$1.00			PN4250	4/$1.00
2N2219A	3/$1.00	PN3567	3/$1.00	2N4400	4/$1.00
2N2221	4/$1.00	PN3568	4/$1.00	2N4401	4/$1.00
2N2222A	5/$1.00	PN3569	4/$1.00	2N4402	4/$1.00
2N2369	5/$1.00	2N3704	5/$1.00	2N4403	4/$1.00
2N2369A	4/$1.00	2N3705	5/$1.00	2N4409	5/$1.00
2N2484	4/$1.00	2N3706	5/$1.00	2N5086	4/$1.00
2N2906A	4/$1.00	2N3707	5/$1.00	2N5087	4/$1.00
2N2907A	5/$1.00	2N3711	5/$1.00	2N5088	4/$1.00
2N2925	5/$1.00	2N3724	$. 65	2N5089	4/$1.00
2N3053	2/$1.00	2N3725	$1.00	2N5129	5/$1.00
2N3055	$ .89	2N3903	5/$1.00	2N5138	5/$1.00
MJE3055	$1.00	2N3904	4/$1.00	2N5139	5/$1.00
MJE2955	$1.25	2N3905	4/$1.00	2N5209	5/$1.00
2N3392	5/$1.00	2N3906	4/$1.00	2N5951	5/$1.00
2N3398	5/$1.00	2N4013	3/$1.00	C106B1SCR	2/$1.00
		2N4014	3/$1.00	2N5432	$2.00
		2N4123	10/$1.00		

CAPACITOR CORNER

50 VOLT CERAMIC DISC CAPACITORS

	1-9	10-49	50-100		1-9	10-49	50-100
10 pf	.05	.04	.03	.001μF	.05	.04	.035
22 pf	.05	.04	.03	.0047μF	.05	.04	.035
47 pf	.05	.04	.03	.01μF	.05	.04	.035
100 pf	.05	.04	.03	.022μF	.06	.05	.04
220 pf	.05	.04	.03	.047μF	.06	.05	.04
470 pf	.05	.04	.035	.1μF	.12	.09	.075
100 VOLT MYLAR FILM CAPACITORS							
.001mf	.12	.10	.07	.022mf	.13	.11	.08
.0022	.12	.10	.07	.047mf	.21	.17	.13
.0047mf	.12	.10	.07	.1mf	.27	.23	.17
.01mf	.12	.10	.07	.22mf	.33	.27	.22
+20% DIPPED TANTALUMS (SOLID) CAPACITORS							
.1/35V	.28	.23	.17	1.5/35V	.30	.26	.21
.15/35V	.28	.23	.17	2.2/25V	.31	.27	.22
.22/35V	.28	.23	.17	3.3/25V	.31	.27	.22
.33/35V	.28	.23	.17	4.7/25V	.32	.28	.23
.47/35V	.28	.23	.17	6.8/25V	.36	.31	.25
.68/35V	.28	.23	.17	10/25V	.40	.35	.29
1.0/35V	.28	.23	.17	15/25V	.63	.50	.40
MINIATURE ALUMINUM ELECTROLYTIC CAPACITORS							
Axial Lead				**Radial Lead**			
.47/50V	.15	.13	.10	.47/25V	.15	.13	.10
1.0/50V	.16	.14	.11	.47/50V	.16	.14	.11
3.3/50V	.15	.13	.10	1.0/16V	.15	.13	.10
4.7/25V	.16	.14	.12	1.0/25V	.16	.14	.11
10/25V	.15	.13	.10	1.0/50V	.16	.14	.11
10/50V	.16	.14	.12	4.7/16V	.15	.13	.10
22/25V	.17	.15	.12	4.7/25V	.15	.13	.10
22/50V	.24	.20	.18	4.7/50V	.16	.14	.11
47/25V	.19	.17	.15	10/16V	.14	.12	.09
47/50V	.25	.21	.19	10/25V	.15	.13	.10
100/25V	.24	.20	.18	10/50V	.16	.14	.12
100/50V	.35	.30	.28	47/50V	.24	.21	.19
220/25V	.32	.28	.25	100/16V	.19	.15	.14
220/50V	.45	.41	.38	100/25V	.24	.20	.18
470/25V	.33	.29	.27	100/50V	.35	.30	.28
1000/16V	.55	.50	.45	220/16V	.23	.17	.16
2200/16V	.70	.62	.55	470/25V	.31	.28	.26

Circle 87 on inquiry card.

computer display terminal

This display terminal has an integral controller, B/W cathode ray tube and keyboard. The system has a serial I/O interface for communication and an I/O interface for a printer

DISPLAY (P/N 4802-1095-501) FEATURES:
- 17" B/W CRT
- 41 lines of data
- 52 characters per line
- Characters are generated by a diode matrix "graphic" technique
- 21 special push-buttons wired for a program call up
- Brightness Control
- Self-contained power supply

KEYBOARD (P/N 4802-1115-501) FEATURES:
- Reed switch technology
- 54 data keys
- 28 special keys detachable with cable

LOGIC UNIT (P/N 4802-1157-502) FEATURES:
- 1024 by 6 bit core memory
- Printer I/O interface
- Communication I/O interface

POWER: 115V, 50/60 Hz, 500 Watts

WEIGHT: 210 lbs. (including logic unit, keyboard, display and cables.)

FOB LYNN MASS (you pay shipping)
Check with order please.



External logic & power pack not shown.

$180.00

"AS IS"

4 way cursor control, graphics display.

The story: These are unused terminals made for airport ticketing & seat assignment. After several years of storage they require tinkering to make operable. We have some hints printed such as cleaning PC fingers. One of our customers has this tied into his KIM-1, another has his running with his IMSAI. We have data on this. Should be useable on most common computers. A hell of a deal and all for a paltry $180.00. Don't be left out as many were on our past VIATRON deal. Sold "as is" all sales final.

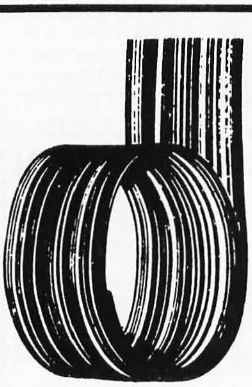
WITH COMPLETE DOCUMENTATION

SPECTRA FLAT TWIST

50 conductor, 28 gauge, 7 strands/conductor made by Spectra. Two conductors are paired & twisted and the flat ribbon made up of 25 pairs to give total of 50 conductor. May be peeled off in pairs if desired. Made twisted to cut down on "cross talk." Ideal for sandwiching PC boards allowing flexibility and working on both sides of the boards. Cost originally $13.00/ft

SP-324-A $1.00/ft. 10 ft/$9.00

SP-234-A $1.00 ft 50 cond. 10 ft/$9.00
SP-234-B .90 ft 32 cond. 10 ft/$8.00



WIRE WRAP WIRE

TEFZEL blue #30 Reg. price $13.28/100 ft. Our price 100 ft $2.00; 500 ft $7.50.

MULTI COLORED SPECTRA WIRE

Footage			10'	50'	100'
8	Cond.	#24	$2.50	9.00	15.00
12	"	22	3.00	11.00	18.00
14	"	22	3.50	13.00	21.00
29	"	22	7.50	28.00	45.00

Great savings as these are about 1/4 book prices. All fresh & new.

In tall TO-5 can DPDT, 24 volts. Brand new. cost $16.00 each

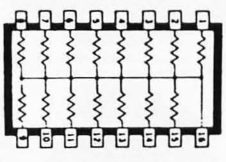
TINY SWITCH TELEDYNE

SP-134 $3.00 each 2/$5.00



Precision 16 pin DIP network as shown. Each resistor 1K. For pull-up/pull-down interface networks. Value over $1.00 each; New, CTS or Beckman

SP-320 pack of 6 $1.00



Meshna

Please add shipping cost on above. Minimum order $10

FREE CATALOG SP-9 NOW READY

P.O. Box 62, E. Lynn, Massachusetts 01904

Circle 18 on inquiry card.

INTERFACE AGE™

MICROCOMPUTING FOR HOME AND THE SMALL BUSINESSMAN

The professional publication bringing microcomputing technology to the hobbyist, small businessman, educator, engineer and student. Every issue edited to bring technology and people together in the simplest manner.

INTERFACE AGE is packed with



- **HARDWARE ARTICLES** — Product profiles, comparisons, applications, modifications and construction projects.
- **SOFTWARE ARTICLES** — Microcomputer development software, short software routines, application software, off-line software storage formats, software communication standards and access to the microcomputer software depository for all **INTERFACE AGE** readers.
- **TUTORIALS** — Fundamentals of micro processors, basics for microcomputing, professional to technical transition information, understanding software, elementary math for computing.
- **NEW PRODUCTS** — Manufacturer profiles and latest product releases.
- **USER COMMUNICATION** — ***UPDATE*** — devoted to club and organization announcements and activities. — ***LETTERS TO THE EDITOR*** — Reader forum for expression of opinions and feedback on articles and features.
- **MICRO-MARKET/FIFO FLEA MARKET** — Low cost/no cost advertising for the new marketeer/garage sale enthusiast.

If you need to know how to get started in microcomputing or need the valuable software once your system is completed then **INTERFACE AGE** is a must for you.

Don't Delay — Subscribe Today!

INTERFACE AGE MAGAZINE

P.O. Box 1234
Cerritos, CA. 90701

12 Monthly Issues:
$10 U.S., $12 Can./Mex.,
$18 International

☐ CHECK ☐ MONEY ORDER

Name ____________________

Address ____________________

City ____________ State ______ Zip ______

Country ____________________

MAKE CHECK PAYABLE TO: **INTERFACE AGE** MAGAZINE

NEW COMPUTER INTERFACE BOARD KIT

$49.00

Our new computer kit allows you to interface serial TTL to RS 232 and RS 232 to TTL. There are four of these supplied with the kit, so you can run up to four devices on one TTL or four separate TTL to RS 232 devices.

Typical use: You can use your computer ports to run an RS 232 printer, video terminal and two other RS 232 devices at once, without constantly connecting and disconnecting your terminals.

Example: Out store to printer — Voltage requirement +5V and ±5V or ±12 V depending on your RS 232 device.

We supply — board, connectors, documentation and components. Sorry, we do not supply case or power supply.

GENERAL PURPOSE COMPUTER POWER SUPPLY KIT

$79.00

This power supply kit features a high frequency torroid transformer with switching transistors in order to save space and weight. 115V 60 cycle primary. The outputs with local regulators are 5V to 10A, in one amp increments, −5V at 1A, ±12V at 1A regulators supplied 6 340T-5 supplied.

UNIVERSAL 4K MEMORY BOARD KIT $74.50

This memory board may be used with the F8 and with minor modifications may be used with KIM-1µp.

32-2102-1 static RAM's, 16 address lines, 8 data lines in, 8 data lines out, all buffered. On-board decoding for any 4 of 64 pages, standard 44 pin, .156″ buss.

F8 EVALUATION BOARD KIT WITH EXPANSION CAPABILITIES

$99.00

A fantastic bargain for only with the following features:

- 20 ma or RS 232 interface
- 64K addressing range
- Program control timers
- 1K of on-board static memory
- Built in clock generator
- 64 Byte register
- Built-in priority interrupts
- Documentation
- Uses Fairbug PSU

FOR FAIRBUG 4K F8 BASIC ON PAPER TAPE $25.00

2708-8K EPROM	$24.95
2522 STATIC SHIFT REG	$ 1.95
2513 CHARACTER GEN	$ 9.95
2518-HEX 32 BIT SR	$ 3.50
2102-1 1024 BT RAM	$ 1.39
5280-4K DYNAMIC RAM	$ 6.95
MM5202A UV PROM	$ 6.95
MM5203 UV PROM	$ 6.95
1702A UV PROM	$ 5.95
5204-4K PROM	$10.95
AY-5-1013 UART	$ 6.95

MINIATURE MULTI-TURN TRIM POTS 100, 500, 2K, 5K, 10K, 25K, 50K, 100K, 200K 1 Meg. $.75 each . . . 3/$2.00

MULTI-TURN TRIM POTS Similar to Bourns 3010 style 3/16″x5/8″x1-1/4″; 50, 100, 1K, 10K, 50K ohms $1.50 ea. . . 3/$4.00

LIGHT ACTIVATED SCR's TO-18, 200V 1A . . . $ 1.75

TRANSISTOR SPECIALS

2N3585 NPN Si TO-66	$ .95
2N3772 NPN Si TO-3	$ 1.60
2N456A PNP GE	$ .75
2N4908 PNP Si TO-3	$ 1.00
2N6056 NPN Si TO-3 Darlington	$ 1.70
2N5086 PNP Si TO-92	4/$ 1.00
2N4898 PNP TO-66	$ .60
2N404 PNP GE TO-5	5/$ 1.00
2N3919 NPN Si TO-3 RF	$ 1.50
MPSA 13 NPN Si TO-92	3/$ 1.00
2N3767 NPN Si TO-66	$ .70
2N2222 NPN Si TO-18	5/$ 1.00
2N3055 NPN Si TO-3	$ .80
2N3904 NPN Si TO-92	5/$ 1.00
2N3906 PNP Si TO-92	5/$ 1.00
2N5296 NPN Si TO-220	$ .50
2N6109 PNP Si TO-220	$ .55
2N3638 PNP Si TO-5	5/$ 1.00
2N6517 NPN TO-92 Si	3/$ 1.00

C/MOS (DIODE CLAMPED)

74C10	.22	4016	.40	4029	1.10
74C193	1.50	4017	1.05	4030	.22
4001	.22	4018	1.00	4033	1.50
4002	.22	4019	.25	4035	1.10
4006	1.20	4020	1.05	4042	.78
4007	.22	4022	.95	4046	2.25
4009	.42	4023	.22	4049	.40
4010	.42	4024	.75	4050	.40
4011	.22	4025	.22	4055	1.50
4012	.22	4026	1.25	4066	.80
4013	.40	4027	.40	4071	.27
4015	.95	4028	.88	4076	1.05

IN 4148 (IN914) . . . 15/$1.00

MCA-81 OPTICAL LIMIT SWITCH $1.50

LED READOUTS

FND 359 C.C. .4″	$ .55	HP 7740-.3″ C.C.	$1.25
FND 70 C.C. .4″	$ .60	MAN-7-.3″ C.A.	$ .95
FND 503 C.C. .5″	$1.05	NS 33-3 dig. array	$ .75
FND 510 C.A. .5″	$1.05	DL 747 C.A. .6″	$1.95

PRINTED CIRCUIT BOARD

4-1/2″x6-1/2″ SINGLE SIDED EPOXY BOARD 1/16″ thick, unetched $.60 ea. . . . 5/$2.60

7 WATT LD-65 LASER DIODE IR $8.95

2N 3820 P FET	$ .45
2N 5457 N FET	$ .45
2N2646	$ .45
ER 900 TRIGGER DIODES	4/ $1.00
2N 6028 PROG. UJT	$ .65
8 PIN DIP SOCKETS	$ .24
14 PIN DIP SOCKETS	$ .25
16 PIN DIP SOCKETS	$ .28
18 PIN DIP SOCKETS	$ .30
24 PIN DIP SOCKETS	$ .40
28 PIN DIP SOCKETS	$ .50
40 PIN DIP SOCKETS	$ .60

VERIPAX PC BOARD

This board is a 1/16″ single sided paper epoxy board, 4½″x6½″ DRILLED and ETCHED which will hold up to 21 single 14 pin IC's or 8, 16, or LSI DIP IC's with busses for power supply connector. . . . $4.00

MV 5691 YELLOW-GREEN BIPOLAR LED	$1.25
FP 100 PHOTO TRANS	$ .50
RED, YELLOW, GREEN or AMBER LARGE LED's	6/$1.00
1L-5 (MCT-2)	$ .75
MOLEX PINS	100/$1.00
	1000/$8.00
10 WATT ZENERS 3.9, 4.7, 5.6, 8.2, 12,15,18,22,100,150 or 200V	ea. $ .60
1 WATT ZENERS 4.7, 5.6, 10, 12, 15 18 or 22V	ea. $ .25
MC6860 MODEM CHIP	$9.95

Silicon Power Rectifiers

PRV	1A	3A	12A	50A	125A
100	.06	.14	.30	.80	3.70
200	.07	.20	.35	1.15	4.25
400	.09	.25	.50	1.40	6.50
600	.11	.30	.70	1.80	8.50
800	.15	.35	.90	2.30	10.50
1000	.20	.45	1.10	2.75	12.50

SILICON SOLAR CELLS

2¼″ diameter

.4V at 500 ma. $4.00 / .2V at 200 mils $2.00

REGULATORS

309K	$ .95	340K-5,12,15 or 24V	$1.25
723	$ .50	340T-5, 6, 8, 12 15,18 or 24V	$1.10
LM 376	$ .60	78 MG	$1.35
320K-5 or 15V	$1.40	79 MG	$1.35
320T-5,12,15 or 24V	$1.25		

RS232 CONNECTORS — DB 25P male . . . $3.25; DB 25S female . . . $3.95

TANTULUM CAPACITORS

.22UF	35V	5/$1.00	6.8UF	35V	3/$1.00
.47UF	35V	5/$1.00	22UF	35V	$ .40
.68UF	35V	5/$1.00	30UF	6V	$ .40
1UF	35V	5/$1.00	33UF	35V	$ .40
2.20UF	20V	5/$1.00	47UF	20V	$ .35
3.3UF	35V	4/$1.00			
4.7UF	15V	5/$1.00			

M7001 ALARM CLOCK CHIP . . . $5.75

NATIONAL MOS DEVICES

MM1402	1.75	MM5057	2.25
MM1403	1.75	MM5058	2.75
MM1404	1.75	MM5060	2.75
MM5013	2.50	MM5061	2.50
MM5016	2.50	MM5555	4.75
MM5017	2.70	MM5556	4.75
MM5055	2.25	MM5210	1.95
MM5056	2.25	MM5260	1.75

#24, EIGHT CONDUCTOR SPECTRA FLAT CABLE 10′/$1.50 100′/$13.50

#30 WIRE WRAP WIRE SINGLE STRAND 100′/$1.40

TTL IC SERIES

7400	.14	7445	.15	74150	.90
7401	.14	7446	.65	74151	.60
7402	.14	7447	.65	74153	.60
7403	.14	7448	.65	74154	.95
7404	.18	7450	.15	74155	.70
7405	.18	7472	.29	74157	.58
7406	.25	7473	.29	74161	.85
7407	.25	7474	.29	74163	.80
7408	.18	7475	.45	74164	.95
7409	.17	7476	.30	74165	.95
7410	.14	7480	.35	74173	1.20
7411	.20	7483	.62	74174	.95
7412	.20	7485	.87	74175	.82
7413	.39	7486	.30	74176	.75
7414	.63	7489	1.85	74177	.75
7416	.25	7490	.42	74180	.65
7417	.25	7491	.58	74181	1.90
7420	.14	7492	.43	74190	1.00
7425	.25	7493	.45	74191	1.00
7426	.22	7494	.70	74192	.83
7427	.25	7495	.65	74193	.83
7430	.14	7496	.65	74194	.85
7432	.25	74107	.28	74195	.52
7437	.21	74121	.33	74257	1.25
7438	.21	74123	.65	74279	.87
7440	.14	74125	.40	75324	1.75
7441	.70	74126	.40	75491	.65
7442	.40	74132	.67	75492	.65

MINIATURE DIP SWITCHES
CTS-206-4 Four SPST switches in one minidip package. . . $1.75
CTS-206-8 Eight SPST switches in a 16 pin DIP package. . . $1.95

5-8V SPST Miniature reed, relay., normally open, 330 Ohm coil resistance. $.75, 3/$2.00

ALCO MINIATURE TOGGLE SWITCHES
MTA 106 SPDT . . . $1.20
MTA 206 DPDT . . . $1.70

Full Wave Bridges

PRV	2A	6A	25A
200	.75	1.25	2.00
400	.95	1.50	3.00
600	1.20	1.75	4.00

SANKEN AUDIO POWER AMPS

Si 1010 G 10 WATTS	$ 7.95
Si 1020 G 20 WATTS	$15.95
Si 1050 G 50 WATTS	$27.95

CCD 110 LINEAR 256 XI BIT SELF SCANNING CHARGED COUPLED DEVICE . . . $65.00

CCD 201 — 100 x 100 CHARGE COUPLED DEVICE . . . $99.00

Send 25¢ for our catalog featuring Transistors and Rectifiers
145 Hampshire St., Cambridge, Mass.

74LS SERIES

74LS00	.23	74LS155	1.40
74LS02	.23	74LS157	.98
74LS04	.28	74LS160	1.50
74LS08	.23	74LS161	1.45
74LS10	.23	74LS162	1.50
74LS11	.23	74LS163	1.50
74LS20	.23	74LS168	2.15
74LS21	.23	74LS169	2.15
74LS22	.23	74LS173	2.45
74LS27	.27	74LS174	1.20
74LS30	.23	74LS175	1.22
74LS32	.33	74LS190	1.75
74LS37	.37	74LS191	1.75
74LS38	.37	74LS192	1.75
74LS74	.49	74LS193	1.75
74LS90	.95	74LS195	1.25
74LS42	.88	74LS196	1.35
74LS73	.43	74LS197	1.35
74LS74	.45	74LS257	1.35
74LS76	.43	74LS258	1.38
74LS90	.89	74LS266	.53
74LS92	.85	74LS365	.68
74LS93	.85	74LS366	.68
74LS109	.50	74LS367	.68
74LS112	.43	74LS368	.68
74LS113	.43	74LS390	2.20
74LS114	.43		
74LS132	1.10		
74LS138	1.25		
74LS139	1.30		
74LS145	1.15		
74LS151	.98		
74LS153	.93		

LINEAR CIRCUITS

LM 101	.75
LM 301/748	.31
LM 307	.30
LM 308	.95
LM 311	.95
LM 318	.35
LM 319	.95
LM 324	.05
LM 339	.10
LM 370	1.15
LM 377	2.50
LM 380	.95
LM 381	1.25
LM 382	1.25
LM 537	2.50
LM 553	2.50
LM 555	.44
LM 556	.85
560	2.00
565	1.10
566	1.50
567	1.50
703	.90
709	.25
710	.35
711	.35
741 C or V	.31
747	.65
LM1310	2.50
1456	.95
1458	.60
CA 3046	.75
CA 3047	.95
3900	.49
3038	3.90

TRIACS / SCR'S

PRV	TRIACS 1A	TRIACS 10A	TRIACS 25A	SCR'S 1.5A	SCR'S 6A	SCR'S 35A
100	.40	.70	1.30	.40	.50	1.20
200	.70	1.10	1.75	.60	.70	1.60
400	1.10	1.60	2.60	1.00	1.20	2.20
600	1.70	2.30	3.60		1.50	3.00

Terms: FOB Cambridge, Mass. Send Check or Money Order. Include Postage, Minimum Order $5.00, COD'S $20.00

SOLID STATE SALES
P.O. BOX 74B
SOMERVILLE, MASS. 02143 TEL. (617) 547-4005

WE SHIP OVER 95% OF OUR ORDERS THE DAY WE RECEIVE THEM

Circle 90 on inquiry card.

Reader Service

To get further information on the products advertised in BYTE, fill out the reader service card with your name and address. Then circle the appropriate numbers for the advertisers you select from this list. Add a 9 cent stamp to the card, then drop it in the mail.

***Correspond directly with company.**

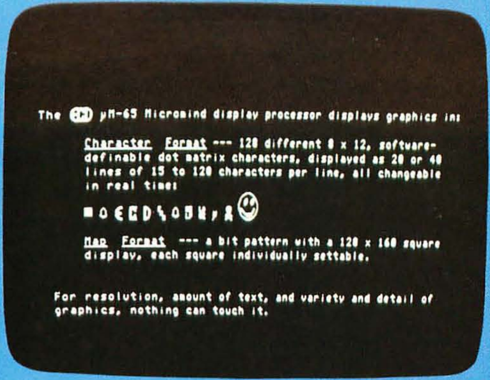
Inquiry No.	Page No.
70	Alpha Digital Systems 163
14	Anderson Jacobson 50
7	Apple 22, 23, 24
11	Backer-Loring 32
51	Bits Inc 129, 135, 142-145, 160
71	Bits and Bytes 163
41	BPI 131
72	Charles Floto 163
73	Cheap Inc 163
56	Comptek 151
4	Compucolor 6, 7
24	Computalker 87
74	Computer Corner 163
75	Computer Depot 163
61	Computer Enterprises 154
76	Computer Hut 163
77	Computer Mart of NH 163
47	Computer Room 137
31	Computer Land 107
68	Computer Store 162
83	Computer Warehouse 167
22	Continental Specialties 83
2	Cromemco 1, 2
39	DAJEN Electronics 128
63	Data Search 155
34	DataSync 113
21	Digital Group 77
33	Digital Research 111
92	ECD CIII
78	Educational Data 164
59	Edityper 153
85	Electronic Warehouse 168
79	EMA Industries 164
27	Extensis 99
86	Godbout 169
*	HAL Communications 134
18	Heath Company 60, 61, 62, 63
28	Heuristics 101
80	Hybird Tech 164
8	iCOM 26, 27
3	IMSAI 5
89	Interface Age 173
50	International Data Systems 141
87	James 170, 171
40	Kent Moore 130
94	Logic Design Inc 164
88	Meshna 172
97	Microcom 129
95	Microcomp 164
58	Mini Computer Suppliers 152
35	Micromation 115, 117
60	Microware 153
44	MSI 133
82	Mikos 166
38	MiniTerm 125, 148
93	MITS CIV
81	Morrow 165
6	Motorola 91
62	mpi 154
55	Mullen 151
29	National Multiplex 103
16	North Star Computers 57
66	Objective Design 160
12	OSI 40
20	OK Tool 76, 139
65	Oliver Audio Engineering 158
43	Omni 132
49	PAIA 140
53	Parasitic Engineering 47, 147
*	PC 77 51,159
*	PerCom Data 11, 150
23	Peripheral Vision 85
*	Personal Computing Expo 161
*	Polymorphic Systems 33
54	Prentice Hall 149
48	Prime Radix 138
5	Processor Technology 8, 9, 10
17	Quay 59
37	RHS Marketing 120, 121
96	Rotundra Cybernetics 164
26	Howard W Sams 97
19	Scelbi 69
*	Scientific Research 35
15	Seals 55
69	Serial Systems 163
32	Smoke Signal Broadcasting 109
57	Software Records 152
13	Solid State Music 41
90	Solid State Sales 174
1	Southwest Tech CII
80	Sunny Computer Stores 164
25	Synchro-Sound Enterprises 88, 89
67	Tarbell Electronics 162
10	Technical Design Labs 31, 117
42	Technical Systems Consultants 132
46	Telpar 136
84	Trico 167
91	Tri Tek 175
52	Urban Instruments 146
9	Vector Graphic 28, 29, 95
36	Worldwide Electronics 116
64	Xybek 155
30	Ximedia 105

BOMB
BYTE's Ongoing Monitor Box

The April BOMB, Exploded...

There are risks associated with every action one can take, and occasionally the results fulfill the worst expectations. The April 1977 issue of BYTE bombed on content with respect to the inclusion of several (ie: four articles) on a fiction theme, and one intentionally outdated (as April Fool) microprocessor update on the 8008 (which became obsolete for new design with the introduction of the Intel 8080). The winner, by a large margin, was the article by Jack Emmerichs on the Tiny Assembler 6800, which came in the $+2\ \sigma$ point relative to the mean for 13 articles. Second place went to a semifiction story by Steve Ciarcia (countering the trend just cited) entitled "Having a Private Affair With Your Computer." Steve's article came in at the $+1\ \sigma$ point, just slightly ahead of a group of articles which clustered above the mean at approximately the same displacement. The remaining fiction articles (and numerous comments on the BOMB cards) clustered around the $-1.5\ \sigma$ point of the observed distribution. These statistics were calculated with a programmable calculator. The results should be taken with a grain of salt, since the standard deviation was 24% of the mean. Why did I put so many of a similar type of article in the April issue? It was, after all, the April Fool issue; and it provided a form of controlled experiment by tweaking the content a little far in one direction while observing the results in the tally of BOMB cards. The experiment has been concluded for the time being . . .C Helmers■

On BOMB Card, Article No.	ARTICLE	PAGE
1	Brown: How to Computerize Your Model Railroad	12
2	Hall: This Circuit Multiplies	36
3	Borrmann: Condensed Reference Chart for the 6800	42
4	Hart-Badger: A Train Control Display Using the LSI-11	44
5	Barbier: The TV Oscilloscope	52
6	Boddie: Speech Recognition for a Personal Computer System	64
7	Maurer: How to Pick up a Dropped Bit	72
8	Grappel. Give Your Micro a Megabyte	78
9	Simmons: An Introduction to Numbers	82
10	Zolman: A Machine Code Relocator for the 8080	92
11	Baker: BASICally BASIC	96
12	Pittet: How Far--Which Way?	118
13	McGahee: Add Cursor Control to Your TVT II	122
14	Grater: Giving KIM Some Fancy Jewels	126

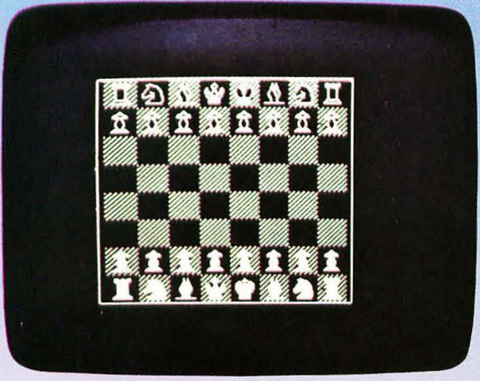


Key Into Maxi-Power @ Micro-Price



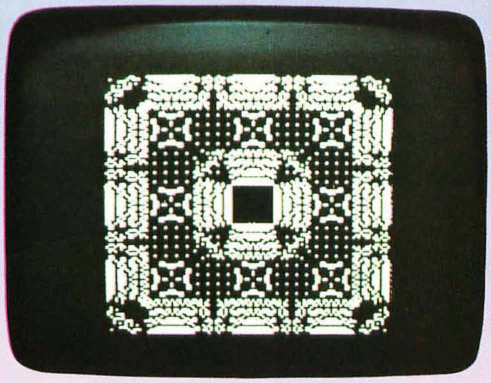
Micromind is an incredibly flexible, complete and expandable, hardware/ software, general purpose computer system. You won't outgrow it.

Hardware includes an 80 key, software-definable keyboard, I/O interface board, 6500A-series microprocessor (powerful enough for advanced computing), a high-detail graphics and character display processor, power supply, rf modulator, and connections for up to 4 tape recorders plus TV or monitor. An interconnect bus permits 15 additional microprocessors, parallel processing and vastly increased computing power.



System software—including ECD's own notsoBASIC high level language, on advanced error-correcting tape cassettes —provides a word processing editor, a powerful assembler, a debugger, a file system, graphic routines, and peripheral handlers. We also include dynamic graphic games: Animated Spacewar and Life.

ECD's standard Micromind μM-65 supplies 8K bytes of memory. Additional 32K byte expansion boards and a mapping option give Micromind expandable access to 64 Megabytes. Utilizing software-controlled I/O channels, Micromind's advanced encoding techniques load data from ordinary tape recorders at 3200 bits per second.



Micromind comes to you ready-to-use, factory assembled and fully tested. Among microcomputers, it has the largest memory capacity and the fastest storage. You're looking at the work of the finest display processor on the market. You won't find a microcomputer with a more powerful CPU. You won't find a computer with a more flexible keyboard. You won't find anything to touch it at **$987.54.**



So, quit the kluge scene and key into Micromind. You'll be a main frame performer, with all the comforts of home. We're not fooling...this is the cat's μ!

ECD CORP.
196 Broadway, Cambridge, Mass. 02139
(617) 661-4400

ECD

Name ____________________
Address ____________________
City/State ____________________ Zip ________
☐ Fantastic! Check enclosed: **$987.54.** Shipping paid by ECD
☐ BankAmericard ☐ Master Charge Mass. Resident add 5% Sales Tax
____________________ Expiration Date ________
Signature ____________________
☐ Send me your brochure.
b

Actual unretouched photographs.

Circle 92 on inquiry card.

/ability.

It Comes Naturally With The Altair™ 8800b

The Altair 8800b from MITS: the second generation design of the microcomputer that started it all. The mainframe that has the abilities everyone is demanding from microcomputers today:

Expand-ability:

The Altair 8800b power supply and one-piece, 18-slot motherboard allow efficient and easy expandability for memory and I/O options. All Altair PC boards are designed to give you maximum capability/lowest power usage possible per board. This means that for each slot used you get more features and require less power, than with any of the "off-brand" Altair-bus-compatible boards.

Whether you buy an entire system up front or choose to expand gradually, it's easy to get the configuration you need with the complete family of Altair peripheral equipment, including floppy disk, line printer, audio cassette record interface, A/D converter, PROM programmer, serial and parallel I/O boards, choice of four different memory boards and many others.

Reli-ability:

The unique design features of the Altair 8800b, which have set the standard for the microcomputer industry, make it the most reliable unit of its kind. The Altair 100-pin bus, the now-standard design used by many imitators, has been "standard" all along at MITS. The unique Front Panel Interface Board on the Altair 8800b isolates and filters front panel noise before it can be transmitted to the bus. The all-new CPU board utilizes the 8080A microprocessor, Intel 8224 clock generator and 8216 bus drivers.

Flex-ability:

Meeting the diversified demands of an ever-increasing microprocessor market requires flexibility: not just hardware flexibility but software flexibility as well. MITS software, including the innovative Altair BASIC language, allows the full potential of the Altair 8800b computer to be realized.

8K ALTAIR BASIC has facilities for variable length strings with LEFT$, RIGHT$, and MID$ functions, a concatenation operator, and VAL AND STR$ functions to convert between strings and numbers.

Extended ALTAIR BASIC allows integer, single and double precision variables, automatic line numbering and renumbering, user-defined string functions, PRINT USING for formatted output and a powerful EDIT command for editing program files during or after entry. Extended statements and commands include IF . . . THEN . . . ELSE, LIST and DELETE program lines, SWAP variables and Trace On and Off for debugging.

Disk ALTAIR BASIC has all the features of Extended BASIC with the additional capability to maintain sequential and random access disk files. Utilities are provided for formatting disks and printing directories.

In all versions of ALTAIR BASIC you get the ease and efficiency of BASIC for the solution of real world problems.

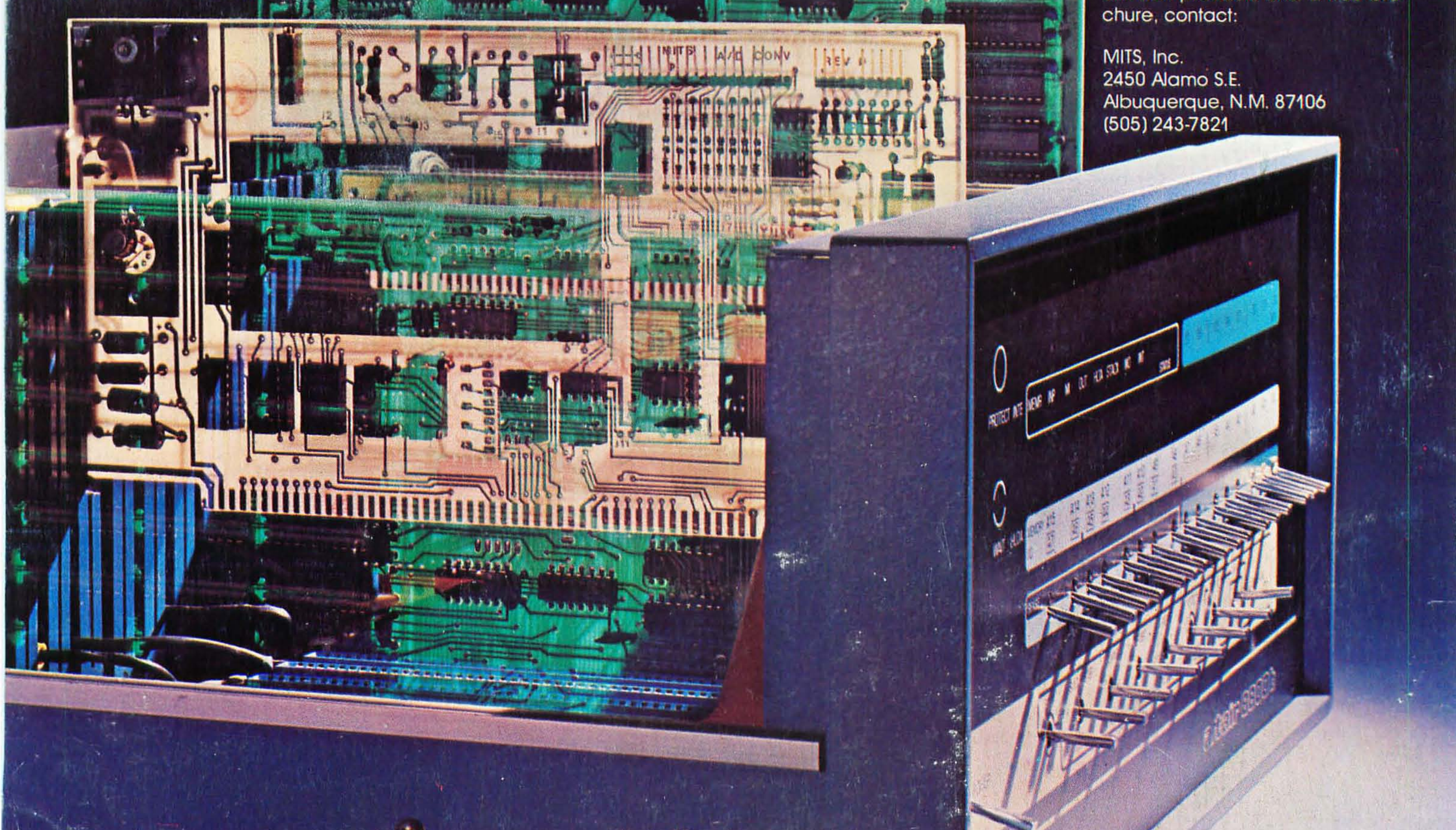
Package II, an assembly language development system for the Altair 8800b, includes system monitor, text editor, assembler and debug.

Afford-ability:

Prices for the Altair 8800b start at $840.00 for a kit and $1100.00 for an assembled unit (all documentation included).

For a complete listing of prices on all Altair products and a free brochure, contact:

MITS, Inc.
2450 Alamo S.E.
Albuquerque, N.M. 87106
(505) 243-7821



NOTE: Altair is a trademark of MITS, Inc.

Circle 93 on inquiry card.